Titre: Title:	Development of an Overset Structured 2D RANS/URANS Navier-Stokes Solver Using an Implicit Space and Non-Linear Frequency Domain Time Operators
Auteur: Author:	Antoine Taillon Lévesque
Date:	2015
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Lévesque, A. T. (2015). Development of an Overset Structured 2D RANS/URANS Navier-Stokes Solver Using an Implicit Space and Non-Linear Frequency Domain Time Operators [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1718/

UNIVERSITÉ DE MONTRÉAL

DEVELOPMENT OF AN OVERSET STRUCTURED 2D RANS/URANS NAVIER-STOKES SOLVER USING AN IMPLICIT SPACE AND NON-LINEAR FREQUENCY DOMAIN TIME OPERATORS

ANTOINE TAILLON LÉVESQUE
DÉPARTEMENT DE GÉNIE MÉCANIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION DU
DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE AÉROSPATIAL)
AVRIL 2015

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

DEVELOPMENT OF AN OVERSET STRUCTURED 2D RANS/URANS NAVIER-STOKES SOLVER USING AN IMPLICIT SPACE AND NON-LINEAR FREQUENCY DOMAIN TIME OPERATORS

présenté par : LÉVESQUE Antoine Taillon
en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. PELLETIER Dominique, Ph. D., président
M. LAURENDEAU Éric, Ph. D., membre et directeur de recherche
M. YANG Hong, Ph. D., membre

ACKNOWLEDGMENTS

I would like to thank most of all my research supervisor, professor Eric Laurendeau. His invaluable guidance, thrust and knowledge allowed me to learn so much during these two years.

I would also like to thank Bombardier Aerospace's Advanced Aerodynamics group for all the in kind support provided.

This project would not have been possible without the financial support of the *Natural Sciences and Engineering Research Council of Canada* (NSERC), the *Fond de Recherche Québécois de Nature et Technologie* (FRQNT) and *Bombardier Aerospace*.

To all my collegues, Thibaut Deloze and Ali Mosahebi for the support and the knowledge I have acquired through your teachings, Alexandre Pigeon and Simon Bourgault-Côté for the countless hours of debbuging and code writting we have made together and to all others for the many discussions and insights that cleared so many roadblocks.

To Sophie and my family, for their unconditional support, interest, motivation, comfort and love.

RÉSUMÉ

Ce projet s'intéresse au développement de méthodes avancées afin de réaliser des simulations numériques en mécanique des fluides. Plus particulièrement, ces méthodes s'appliqueront aux modèles *RANS* et *URANS* sur des profils aérodynamiques simples et multi-éléments. Ces développements seront utilisés afin de réaliser une optimisation sur l'interstice et le chevauchement d'un volet de bord de fuite ainsi que des simulations instationnaires standard.

Le logiciel utilisé comme plateforme de développement est *NSCODE*, un solveur Navier-Stokes bidimensionnel pour maillages structurés. Les développements logiciels seront réalisés dans un cadre rigoureux et en utilisant des techniques de programmations appropriées. Les méthodes implémentées viseront plusieurs aspects du solveur incluant les capacités topologiques, l'opérateur spatial et l'opérateur temporel. Afin de traiter les profils multi-éléments, la méthode multi-blocs sera implémentée afin de partitionner le domaine de calcul. La méthode chimère est ensuite implémentée afin d'améliorer la flexibilité de la méthode multi-blocs. Le schéma de dissipation artificielle scalaire est ensuite remplacé par le schéma de dissipation matricielle afin d'améliorer la précision du solveur. Un schéma d'opérateur spatial utilisant un préconditionneur Jacobien implicite par point ainsi qu'un schéma implicite *Block Lower-Upper Symmetric Gauss Seidel* (LU-SGS) sont ensuite implémentés afin d'améliorer le taux de convergence du solveur. L'opérateur temporel par pas de temps double présent dans le logiciel initial est adapté afin d'être compatible avec les différents schémas d'opérateurs spatiaux utilisés. Un opérateur temporel Non-Linéaire dans le Domaine Fréquentiel (NLFD) est ensuite implémenté afin de résoudre efficacement les écoulements instationnaires périodiques.

Chacune des méthodes implémentées est validée et vérifiée en utilisant des cas tests utilisés dans la littérature ainsi qu'avec des résultats expérimentaux. Une grande variété de cas tests sont utilisés afin de s'assurer de la fiabilité du solveur lors des applications futures.

Les implémentations logicielles sont ensuite utilisés afin de résoudre deux problèmes:

- Une optimisation de dispositif hypersustentateur;
- Une simulation dans le domaine fréquentiel d'un profil aérodynamique en tangage dans un écoulement turbulent.

L'optimisation du dispositif hypersustentateur vise à maximiser le coefficient de portance du profil de recherche *MDA* en modifiant la position du volet de bord de fuite. En utilisant une optimisation utilisant des simulations purement bidimensionnelles en parallèle à une

optimisation utilisant des simulations tridimensionnelles utilisant l'hypothèse d'aile en flèche infinie, la démonstration est faite qu'une optimisation bidimensionnelle n'est pas adaptée au design de dispositifs hypersustentateurs sur des ailes en flèches. La seconde application a pour but d'introduire un modèle de turbulence aux simulations *NLFD* dans *NSCODE*. En tant qu'étape vers l'utilisation de modèles de turbulence à une et deux équations, un modèle algébrique est utilisé. Ce problème vérifiera donc l'utilisation d'un modèle de turbulence algébrique sur un opérateur NLFD. La simulation *NLFD* d'un profil en tangage dans un écoulement turbulent donne des résultats en accord avec la littérature et avec les simulations par pas de temps double ce qui ouvre la voie à l'utilisation de modèles de turbulence plus complexes avec la méthode NLFD.

ABSTRACT

This project aims at performing 2D RANS and URANS computational fluid dynamics simulations over single and multi-element airfoil. The application of such developments is demonstrated via flap gap/overlap optimisation and standard URANS cases.

The work presented in this thesis was implemented in NSCODE, a 2D structured grid Reynolds-Averaged Navier-Stokes flow solver, and is included in a solid framework to ensure its quality and maintainability. It covers many aspects of the flow solver, including topology capabilities, steady and unsteady solver schemes. To simulate flows around complex geometries, the multi block technique is implemented in order to partition the computationnal domain. The multi block capability is then expanded to overset meshes with the Chimera method to allow for even more flexibility in geometry treatment. The existing scalar dissipation scheme is replaced by the matricial artificial dissipation scheme (MATD) to increase spatial resolution accuracy. A point implicit Point-Jacobi Preconditioner and an implicit *Block Lower-Upper Symmetric Gauss Seidel* (LU-SGS) space solving scheme are then implemented to increase convergence rates. The time discretization schemes are also improved. The baseline dual time stepping scheme is modified to be compatible with the LU-SGS solver schemes. A *Non-Linear Frequency Domain* is also added to the software in order to efficiently solve periodic problems.

Each of these techniques is validated and verified against literature data and experimental data. A wide range of test case is chosen in order to ensure full confidence in the developped software.

The software capability developments are then used to solve two problems:

- A high-lift airfoil optimisation;
- An unsteady simulation of turbulent flows in the frequency domain of a pitching airfoil.

The high-lift airfoil optimisation seeks to maximise the lift coefficient of the McDonnel Douglas Research airfoil by changing the flap's position. Using a two dimensional approach in parallel to a three dimensional approach with infinite swept wing hypothesis, a physical phenomenon that couldn't be previously observed on two dimensional solvers was captured. The second case sought to introduce a turbulence component to the NLFD implementation in NSCODE. As a step before using one and two equations turbulence models, an algebraic turbulence model is used in the study. This problem will thus test the applicability of an algebraic turbulence model to the NLFD method. The NLFD resolution of a turbulent pitching

airfoil yielded results that validated very well with the literature and equivalent Dual Time Stepping resolutions, paving the way for the use of more complex turbulence models in NLFD resolutions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS AND ABREVIATIONS

ALE	Arbitrary Lagrangian Eulerian
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy number
DTS	Dual Time Stepping
FFT	Fast Fourrier Transform
FFTW	Fastest Fourrier Transform in the West
GCL	Geometric Conservative Law
IRS	Implicit Residual Smoothing
MATD	Matrix Artificial Dissipation
MDA	McDonnell Douglas research Airfoil
NLFD	Non-Linear Frequency Domain
NS	Navier-Stokes
PJ	Point Jacobi
RANS	Reynolds Averaged Navier-Stokes
TSL	Thin Shear Layer
URANS	Unsteady Reynolds Averaged Navier-Stokes

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

For more than fifty years, the rapid development of computational fluid dynamics (CFD) has completely changed the way aircrafts are designed. The advancements in fluid dynamics physics understanding and the availability of exponentially growing computing power allowed CFD to become a major analysis tool for aerodynamic design. More and more, CFD simulations tend to be used alongside wind tunnel testing due to their increased accuracy and reduced costs. Although numerical simulations via Reynolds-Averaged Navier-Stokes (RANS) solvers are regarded as adequate tools for cruise design conditions, further developments are still required to accurately and efficiently compute flows over high-lift devices as well as unsteady flows. New methods are still being developed and investigated to resolve these issues in order to widen the use of CFD in the aircraft design process.

1.1 Basic Concepts

1.1.1 Navier-Stokes Equations

The CFD methods examined in this work are solving the Reynolds-Averaged Navier-Stokes equations. These equations consists of transport equations solving conservation of mass, momentum and energy in the fluid. The general formulation of these equations in three dimensions is (Versteeg and Malalasekera, 2007)

$$\begin{aligned}
\frac{\partial \rho}{\partial t} + div\left(\rho \vec{u}\right) &= 0 \\
\frac{\partial}{\partial t}\left(\rho u\right) + div\left(\rho \vec{u} u\right) &= -\frac{\partial p}{\partial x} + div\left(\mu \ \ grad(u)\right) + S_{Mx} \\
\frac{\partial}{\partial t}\left(\rho v\right) + div\left(\rho \vec{u} v\right) &= -\frac{\partial p}{\partial y} + div\left(\mu \ \ grad(v)\right) + S_{My} \\
\frac{\partial}{\partial t}\left(\rho w\right) + div\left(\rho \vec{u} w\right) &= -\frac{\partial p}{\partial x} + div\left(\mu \ \ grad(w)\right) + S_{Mz} \\
\frac{\partial}{\partial t}\left(\rho E\right) + div\left(\rho \vec{u} E\right) &= -p div(\vec{u}) + div\left(k \ \ grad(T)\right) + \Phi + S_E
\end{aligned} \tag{1.1}$$

where μ the viscosity coefficient and k the heat conductivity are fluid properties, ρ is the density, u, v, w are the cartesian components of the fluid velocity $\vec{u}$. The source terms S represent other forces and energy sources acting on the fluid such as plasma actuators or gravity.

Equations 1.1 can be rewritten as a general transport equation using ϕ as a general conservative variable

$$\frac{\partial(\rho\phi)}{\partial t} + div(\rho\vec{u}\phi) = div(\Gamma \ grad(\phi)) + S_\phi \tag{1.2}$$

with Γ the appropriate fluid property depending on the equation. Equation 1.2 is solved numerically using space and time discretisation operators.

1.1.2 Finite Volume

In CFD applications, three main space discretisation techniques are used to solve equations 1.2: the finite volume method, the finite difference method and the finite element method. The finite volume method is the most widely used on modern CFD software within the aeronautical industry due to its ability to naturally model transport equations.

The principle of the method is to integrate equation 1.2 on a control volume to yield

$$\frac{\partial}{\partial t}\int_{CV}\rho\phi dCV = -\int_A \vec{n}\cdot(\rho\vec{u}\phi)\,dA + \int_A \vec{n}\cdot(\Gamma \ grad(\phi))\,dA + \int_{CV} S_\phi dV \tag{1.3}$$

Where CV represent the control volume and A the boundaries of the volumes. A physical interpretation of equation 1.3 is that the rate of change of the variable ϕ in the control volume is equal to the flux of ϕ passing through the boundaries of the control volume, plus the diffusion of ϕ through the boundaries, and the source of ϕ within the volume. Discretization in time determines the physical time step Δt. In the case where the steady state solution is sought, a pseudo-time step is defined based on stability bounds instead of the physics of the problem. The right hand side of equation 1.3 is often called residual and is treated as convective residual $\int_A n\cdot(\rho\vec{u}\phi)\,dA$, viscous residual $\int_A n\cdot(\Gamma \ grad(\phi))\,dA$ and source terms residuals $\int_{CV} S_\phi dV$. Convergence is obtained when the solution leads to a residual of zero.

1.1.3 Meshing

Often undervalued in flow simulations, the mesh is a very important component of any CFD methods. Mesh quality is critical to capture the desired physical phenomena as well as obtaining fast solution convergence. The role of the mesh, also called grid, is to cover all the computational domain with nodes on which the space discretisation applies. For the purpose of this work, only conformal meshes will be treated. Those are the meshes whose boundaries coincide with the computational domain boundaries.

Meshes are classified into two main categories, structured and unstructured grid. Structured grids are Cartesian grid in the topological space transformed to conform to a specific domain.

In a structured grid, each node can be identified via its topological coordinates i, j, k for three dimensional meshes or i, j for two dimensional meshes. This property allows for much easier treatment (Blazek, 2005) since the neighbours of each node can be found quickly. Unstructured grids are not arranged in this manner in the topological space and additional information must be stored to find the neighbours of each nodes. The constraint of organising the grid in topological space however leads to difficulties treating complex domains. In external aerodynamics, many simulations have complex geometries involving more than one body such as wing-aileron configuration, wing-flap configuration, wing-fuselage configuration, etc.

Structured meshes are divided into two main sub-categories, single block and multiblock grids. Single block grids are the simplest form of structured meshes and are simply a Cartesian grid in the topological domain deformed to adapt to a specific computational domain. Multiblock meshes can be seen as an assembly of several blocks where the domain is subdivided into sections, each of the section being covered with a different single block mesh. There are different types of multiblock meshes depending on the constraints on block boundaries. One of the most used multiblock mesh type is the *one to one* where different blocks have the same points distributions along their common boundaries, providing continuity in the mesh at the block boundaries. This property of *one to one* meshes allow for easier connection treatment between blocks. *Sliding meshes* are similar to *one to one* except that the node distribution on either side of boundaries are not identical leading to additional work to ensure flow conservation at the block boundaries.

A very interesting multiblock mesh type is the overset meshes, also called chimera meshes. The main difference of overset meshes compared to other multiblock grids is that the blocks are not required to be adjacent to one another. Indeed, blocks can overlap each other arbitrarily in this type of meshes, removing topological constraints during grid creation compared to multiblock meshes. Communication between overset blocks are normally assured via interpolations and involve more work than structured multiblock meshes. The computation required to set up chimera meshes for the solver are typically done in a dedicated preprocessor (Suhs *et al.*, 2002).

1.1.4 Unsteady simulations

Unsteady simulations are required at many stages in aircraft design. Gas turbine, rotor craft flights and aircraft stability are domains where the phenomena observed are changing over time. Two main types of technique are used to solve these problems, time integration and spectral methods.

The time integration technique advances the solution in time via time steps. Using different methods, these techniques discretise the time derivative on the left hand side of equation 1.2 to simulate any type of unsteady phenomena. Spectral methods, on the other hand, express the quantities in equation 1.3 in terms of harmonic functions. Because of the nature of the harmonic functions they use, spectral methods are however limited to periodic flows.

1.2 Elements of the Problematics

1.2.1 Computational Costs of Navier-Stokes Simulations

Currently, the main element limiting the use of CFD in the industry is the computational cost of the simulations. Lower simulation time would enable more simulations to be computed and thus reaching better designs through higher number of optimisation iterations and variables.

Simply increasing the available computing power would achieve this objective. However, supercomputer installation is a process with its own challenges. Apart from the cost of the hardware, extensive infrastructures are needed around the supercomputer, such as cooling, security, energy, etc. While the decreasing costs of computational resources means that the available computing power for the industry is bound to increase over time, work must also be made to find techniques that will solve the Navier-Stokes equations with reduced computational resources.

Efficient spatial operators that would reduce the number of iterations needed to obtain convergence of a solution is an avenue to reduce the computationnal costs of a simulation, provided that the upgraded spatial operator does not lose in iteration's cost what it saves in iteration number. Time operators are also critical. Since unsteady simulations contain one more dimension, time, than steady simulations, their computational costs are significantly higher. Exceptional costs savings can thus be obtained by using more efficient time operators.

1.2.2 Accurate Solutions of numerical Navier-Stokes Simulation

Accuracy of CFD simulations is also an important issue. Discretisation, artificial dissipation and turbulence modelling all introduce different error sources in numerical flow simulations.

Wind tunnel testing and flight testing are additional experimental tools available to aerodynamisists. These experimental tools also have their own limitations: wind-tunnel walls, intrusive measures, Reynolds number limit, aero-elastic deformations, etc. They also carry significant costs. Any accuracy improvement in computational technique would thus lead to significant costs saving on aircraft design programs.

1.2.3 Complex Geometry Treatment

As mentioned previously, space discretisation is a source of error in computational simulations. This becomes especially true when simulating high lift configurations containing multiple bodies (Rumsey *et al.*, 2011). Structured meshes particularly struggle with complex geometries due to their topology constraints.

1.3 Objectives

The current work focuses on developing a framework that will efficiently tackle steady and unsteady flows over geometrically complex airfoils shapes. The framework is *NSCODE*, a flow solver developed by professor Laurendeau's research group as a platform to develop novel numerical methods. Making use of state of the art acceleration techniques, the capabilities of *NSCODE* will be expanded in three main avenues:

- Implement techniques to handle complex geometries;
- Implement efficient steady solver schemes;
- Implement efficient unsteady solver schemes.

1.4 Plan of Thesis

The thesis is split into three main sections: literature review, software development and numerical results.

As the first section, the literature review covers the main techniques used to obtain efficient spatial and time operators. Different governing equations will be derived and explained as well as their associated techniques.

The second section, containing the bulk of the work, details the developments of *NSCODE* conducted within the scope of this work. The framework foundations of *NSCODE* will be discussed as they are key to an efficient and lasting software, followed by the developments in topological capabilities made to treat complex geometries. Next, the improvements to the steady solver will cover improvements to dissipation scheme accuracy and spatial operator. The section ends with unsteady flow solver improvements in both time integration and spectral methods. Validation and verification phases for each development are also included in the section.

The last section will contain some challenging CFD simulations made with *NSCODE* using the developments described in the previous section. First, a high-lift optimisation study is made on the McDonnel Douglas Research Airfoil (MDA) in order to find the optimal flap position for maximum lift coefficient. Then, a pitching airfoil case under turbulent flow will be simulated using the different time operators previously developed in *NSCODE*.

CHAPTER 2 LITERATURE REVIEW

This literature review aims to set the theoretical fundamentals of fluid dynamics concepts covered in this thesis. The first part will focus on developments concerning steady finite volume solvers for the Euler and Navier-Stokes equations. The second part will focus on unsteady flow solvers and is subdivided into two sections. Time-domain solvers advancing the solution in time will first be discussed, followed by frequency domain solvers used for periodic flows.

2.1 Steady Flow Solver

The Navier-Stokes equations, defining the behaviour of viscous flow fields may be expressed in their integral form as

$$\frac{\partial}{\partial t}\int_{\Omega}\vec{W}d\Omega + \oint_{\partial\Omega}(\vec{F_C} - \vec{F_V})dS = 0 \tag{2.1}$$

over the domain Ω with a boundary $\partial\Omega$. The conservative variables vector is, in two dimensions,

$$\vec{W} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix} \tag{2.2}$$

the convective fluxes are

$$\vec{F_C} = \begin{bmatrix} \rho V \\ \rho u V + n_x p \\ \rho v V + n_y p \\ \rho H V \end{bmatrix} \tag{2.3}$$

and the viscous fluxes are

$$\vec{F_V} = \begin{bmatrix} 0 \\ n_x\tau_{xx} + n_y\tau_{xy} \\ n_x\tau_{yx} + n_y\tau_{yy} \\ n_x\Theta_x + n_y\Theta_y \end{bmatrix} \tag{2.4}$$

where

$$\begin{aligned} \Theta_x &= u\tau_{xx} + v\tau_{xy} + k\frac{\partial T}{\partial x} \\ \Theta_y &= u\tau_{yx} + v\tau_{yy} + k\frac{\partial T}{\partial y} \end{aligned} \tag{2.5}$$

$$\tag{2.6}$$

n_x and n_y are the components of outward unit normal vectors to domain boundaries, V is the contravariant velocity and H the total energy. The pressure p is defined with the perfect gas law

$$p = (\gamma - 1)\,\rho\left(E - \frac{1}{2}\left(u_i^2\right)\right) \tag{2.7}$$

with γ the ratio of specific heats, and the stagnation enthalpy by

$$H = E + \frac{p}{\rho} \tag{2.8}$$

Defining the residual as

$$R\,(W) = \oint_{\partial\Omega} (\vec{F_C} - \vec{F_V}) dS \tag{2.9}$$

equation 2.1 can be approximated as

$$\Omega\frac{dW}{dt} + R\,(W) = 0 \tag{2.10}$$

An implicit system can be obtained from 2.10 by using a backward Euler difference and linearising around the time level n

$$\left(\Omega\frac{I}{\Delta t} + \frac{\partial \bar{R}}{\partial W}\right)\delta W^n = -R\,(W^n) \tag{2.11}$$

Where $\bar{R}$ is an approximation of the residual R.

$\frac{\partial \bar{R}}{\partial W}$ can be assembled from the first order approximations of convective and viscous fluxes contribution of a cell and its adjacent neighbours. The upwind flux of Roe(Roe, 1981) are used as a first-order equivalent of the artificial dissipation terms with A and $|A|$ as the convective flux Jacobian and absolute flux Jacobian respectively

$$F^{c^-}_{i+1/2} = \frac{1}{2}(F_{i+1} + F_i) - \frac{1}{2}|A|_{i+1/2}(W_{i+1} - W_i) \tag{2.12}$$

The unidimensional notation is used for simplicity. The viscous fluxes are similarly assembled

using the thin shear layer (TSL) approximation of the viscous flux Jacobian A^v which states that the variations in the flow following computing directions parallel a wall boundary are negligible compared to the variations normal to the wall.

$$F^{v^-}_{i+1/2} = -A^v_{i+1/2}W_{i+1} + A^v_{i+1/2}W_i \tag{2.13}$$

Equation 2.11 diagonal, upper and lower block Jacobians are thus

$$\frac{\partial \bar{R}_i}{\partial W_i} = [D] = +\frac{1}{2}|A|_{i+1/2} + \frac{1}{2}|A|_{i-1/2} + A^v_{i+1/2} + A^v_{i-1/2} \tag{2.14a}$$

$$\frac{\partial \bar{R}_i}{\partial W_{i+1}} = [U] = -\frac{1}{2}A_{i+1} - \frac{1}{2}|A|_{i+1/2} - A^v_{i+1/2} \tag{2.14b}$$

$$\frac{\partial \bar{R}_i}{\partial W_{i-1}} = [L] = -\frac{1}{2}A_{i-1} - \frac{1}{2}|A|_{i-1/2} - A^v_{i-1/2} \tag{2.14c}$$

The one dimensional formulation in the i direction is used for simplicity, $\bar{R}$ is evaluated by the summation over all computational directions.

2.1.1 Scalar Dissipation

In order to prevent odd-even decoupling and oscillation in regions of strong pressure gradient, it is necessary to add artificial dissipation terms to the finite volume scheme as shown by Jameson *et al.* (1981). From 2.12, the absolute flux Jacobian can be expressed as $|A| = T|\Lambda|T^{-1}$. The scalar dissipation scheme (JST) replaces each eigenvalues by the spectral radius multiplied by Martinelli's (Martinelli, 1987) scaling. The absolute flux Jacobian matrix in equation 2.12 is thus effectively replaced by a scalar value

$$F^{c^-}_{i+1/2} = \frac{1}{2}(F_{i+1} + F_i) - \frac{1}{2}\lambda\left(A_{i+1/2}\right)(W_{i+1} - W_i) \tag{2.15}$$

where $\lambda\left(A_{i+1/2}\right)$ is the spectral radius of the convective flux Jacobian at the cell face.

Instead of the second difference described in 2.15, Jameson *et al.* (1981) described an artificial dissipation scheme using a blend of second and fourth difference

$$\epsilon^{(2)}_{i+\frac{1}{2}}(W_{i+1} - W_i) - \epsilon^{(4)}_{i+\frac{1}{2}}(W_{i+2} - 3W_{i+1} + 3W_i - W_{i-1}) \tag{2.16}$$

Where $\epsilon^{(2)}$ and $\epsilon^{(4)}$ coefficients are adapted to the flow using a pressure switch

$$\nu_i = \frac{|p_{i+1} - 2p_i + p_{i-1}|}{|p_{i+1}| + 2|p_i| + |p_{i-1}|} \tag{2.17}$$

The switch is used to detect strong pressure gradients, such as in shockwaves, to increase the contribution of the second difference scheme. The ϵ coefficients are then found using

$$\epsilon^{(2)}_{i+\frac{1}{2}} = \kappa^{(2)} max\left(\nu_{i+1}, \nu_i\right) \tag{2.18a}$$

$$\epsilon^{(4)}_{i+\frac{1}{2}} = max\left(0, \left(\kappa^{(4)} - \epsilon^{(2)}_{i+\frac{1}{2}}\right)\right) \tag{2.18b}$$

$\kappa^{(2)}$ and $\kappa^{(4)}$ are the artificial dissipation coefficients and typically take the values of $\frac{1}{4}$ and $\frac{1}{128}$ for the scalar dissipation scheme.

2.1.2 Matrix Dissipation

While the scalar dissipation scheme provides a simple and fast means to apply artificial dissipation, keeping the absolute flux Jacobian's matrix formulation provides a more accurate scheme. Matricial dissipation scheme (MATD) scales each equation with the appropriate eigenvalues rather than the spectral radius (Swanson and Turkel, 1998),

Using the convective flux Jacobian however leads to numerical difficulties. Near stagnation points and sonic lines, some eigenvalues approach zero. In practice, eigenvalues are thus limited to a fraction of the spectral radius as follows

$$\begin{aligned} |\lambda_1| &= \phi\ max\left(|V + \sqrt{a_1^2 + a_2^2}c|, V_n\rho(A)\right), \\ |\lambda_2| &= \phi\ max\left(|V - \sqrt{a_1^2 + a_2^2}c|, V_n\rho(A)\right), \\ |\lambda_3| &= \phi\ max\left(|V|, V_l\rho(A)\right), \end{aligned} \tag{2.19}$$

where

$$\begin{aligned} a_1 &= J^{-1}\xi_x \\ a_2 &= J^{-1}\xi_y \\ V &= a_1 u + a_1 v \end{aligned} \tag{2.20}$$

and ϕ is the Martinelli's scaling (Martinelli, 1987) to account for grid aspect ratio, which is

in the **I** direction

$$\phi_i = 1 + \left(\frac{\rho\left(A_j\right)}{\rho\left(A_i\right)} \right)^{\beta} \tag{2.21}$$

where β is typically set to 0.5.

The V_n and V_l are determined numerically. Typically, values of 0.2 are used and are increased to 0.4 on coarse grids when using multigrid technique for more robustness.

This yields the eigenvalue vector

$$\Lambda \;=\; Diag[\lambda_1 \quad \lambda_2 \quad \lambda_3 \quad \lambda_3] \tag{2.22}$$

Which is used to recover the absolute convective flux Jacobian matrix via the following equations (Swanson and Turkel, 1997)

$$|A| = |\lambda_3| I + \left(\frac{|\lambda_1| + |\lambda_2|}{2} - |\lambda_3| \right) \left(\frac{\gamma - 1}{c^2} E_1 + \frac{1}{a_1^2 + a_2^2} E_2 \right) + \frac{|\lambda_1| + |\lambda_2|}{2} \left(\frac{1}{\sqrt{a_1^2 + a_2^2} c} \right) \left(E_3 + (\gamma - 1) E_4 \right) \tag{2.23}$$

$$E_1 \;=\; \begin{bmatrix} \phi & -u & -v & 1 \\ u\phi & -u^2 & -uv & u \\ v\phi & -uv & -v^2 & v \\ H\phi & -uH & -vH & H \end{bmatrix}$$

$$E_2 \;=\; \begin{bmatrix} 0 & 0 & 0 & 0 \\ -a_1 q & -a_1^2 & a_1 a_2 & 0 \\ -a_2 q & a_1 a_2 & -a_2^2 & 0 \\ -q^2 & q a_1 & q a_2 & 0 \end{bmatrix}$$

$$E_3 \;=\; \begin{bmatrix} q & a_1 & a_2 & 0 \\ -uq & u a_1 & u a_2 & 0 \\ -vq & v a_1 & v a_2 & 0 \\ -Hq & H a_1 & H a_2 & 0 \end{bmatrix}$$

$$E_4 \;=\; \begin{bmatrix} 0 & 0 & 0 & 0 \\ a_1\phi & -a_1 u & -a_1 v & a_1 \\ a_2\phi & -a_2 u & -a_2 v & a_2 \\ q\phi & -qu & -qv & q \end{bmatrix}$$

$$\phi \;=\; \left(u^2 + v^2 \right) / 2$$

2.1.3 Runge-Kutta Time Integration

A widely used technique to integrate equation 2.10 in time is to utilise an explicit multi stage Runge-Kutta scheme. Jameson (Jameson, 1991) developed an efficient 5 stage scheme taking the following form

$$\begin{aligned} W^{n,0} &= W^{n-1,K} \\ W^{n,k+1} &= W^{n,0} - \alpha_k \frac{\Delta t}{\Omega} \hat{R}\left(W^{n,k}\right), \qquad where \quad k = 0, K-1 \\ W^{n+1,0} &= W^{n,K} \end{aligned} \tag{2.24}$$

Where the residual is modified as follows (Jameson, 1983)

$$\hat{R}\left(W^{n,k}\right) = R^c\left(W^{n,k}\right) + R^v\left(W^{n,k}\right) + \beta_k R^d\left(W^{n,k}\right) + \left(1-\beta_k\right) R^d\left(W^{n,k-1}\right) \tag{2.25}$$

R^c, R^v and R^d being respectively the convective, viscous and dissipative parts of the residual. The α and β stage coefficients used are based on Martinelli's work (Martinelli, 1987)

$$\alpha_k = (1/4 \;\; 1/6 \;\; 3/8 \;\; 1/2 \;\; 1 \;\;) \tag{2.26}$$

$$\beta_k = (\;\; 1 \;\;\; 0 \;\;\; 0.56 \;\;\; 0 \;\;\; 0.44) \tag{2.27}$$

To ensure stability, timesteps are based on Courant-Friedrichs-Lewy (CFL) number.

Allmaras (Allmaras, 1993) and Pierce et al(Pierce and Giles, 1997), improved the explicit Runge-Kutta integration via the addition of a point-Jacobi preconditionner. The Runge-Kutta stages can be rewritten as

$$W^{k+1} = W^0 - \alpha_k \omega [D]^{-1} \hat{R}\left(W_k\right), \qquad where \quad k = 0, K-1 \tag{2.28}$$

where ω is the allowable CFL number.

2.1.4 Implicit Residual smoothing

The Implicit Residual Smoothing (IRS) (Jameson and Baker, 1983) method aims at improving the maximum allowable time-step or CFL number by using an implicit operator on the residuals of explicit schemes. The method requires to solve a tridiagonal matrix for each residual of the conservative variable on each computational directions using the following

formulation

$$-\epsilon_I \vec{R}^*_{I-1} + (1+2\epsilon_I)\,\vec{R}^*_I - \epsilon_I \vec{R}^*_{I+1} = \epsilon_I \vec{R}_{I-1} \tag{2.29}$$

with R^* the smoothed residual. ϵ is defined by Swanson and Turkel (1997) as

$$\epsilon_I = max\left(\frac{1}{4}\left(\left(\frac{N}{N^*}\frac{\phi}{1+r_I}\right)^2 - 1\right), 0\right) \tag{2.30}$$

where ϕ is the same coeficient used for Martinelli's scaling (see equation 2.21), r_I is the ratio of spectral radii

$$r_I = \frac{\lambda_J}{\lambda_I} \tag{2.31}$$

and $\frac{N}{N^*}$ is given the value 2.0.

Using a tridiagonal solver on equation 2.29, the method is computationally inexpensive and increases the maximum allowable CFL number.

2.1.5 Block lower-upper symmetric Gauss-Seidel scheme

Yoon and Jameson(Yoon and Jameson, 1986; Jameson and Yoon, 1987; Yoon and Jameson, 1988) developed a lower-upper symmetric Gauss-Seidel (LU-SGS) scheme to solve equation 2.11. The scheme relies on a forward and backward sweep through the domain

$$[D]\delta W^1_i = -R\left(W^n\right)_i - [L]\delta W^1_{i-1} \tag{2.32a}$$

$$[D]\delta W^2_i = -R\left(W^n\right)_i - [L]\delta W^1_{i-1} - [U]\delta W^2_{i+1} \tag{2.32b}$$

Equation 2.32 uses the full matrix operators defined in 2.14 to enhance convergence rate as described in Wright *et al.* (1996). In the first sweep, the δW information previously computed within the sweep on other cells are used with the $[L]$ block. Second sweep uses δW of previously computed cells of the second sweep with the $[U]$ block and the informations of the previous sweep for the cells not yet computed on the current sweep for the $[L]$ block. The state vector is updated once the sweeps are completed

$$W^{k+1} = W^k + \omega\delta W \tag{2.33}$$

Where ω is the LU-SGS relaxation factor typically ranging between 0.5 and 1.0 (Cagnone *et al.*, 2011).

It is possible to increase the number of sweeps in order to reduce the block tridiagonal matrix inversion errors as shown in Cagnone *et al.* (2011). This is done by alternating between forward and backward sweeps using the latest available solution for δW. For example, the third and fourth sweeps operations would be

$$[D]\delta W_i^3 = -R\left(W^n\right)_i - [U]\delta W_{i+1}^2 - [L]\delta W_{i-1}^3 \tag{2.34a}$$

$$[D]\delta W_i^4 = -R\left(W^n\right)_i - [L]\delta W_{i-1}^3 - [U]\delta W_{i+1}^4 \tag{2.34b}$$

2.1.6 Multigrid

Probably the most effective convergence acceleration method for subsonic and transonic aerodynamic CFD simulations is the multigrid method. Applied to CFD by South Jr and Brandt (1976) and further developed by Jameson (1986), the method is now used in many CFD solvers. However, its implementation is difficult and optimal convergence rates are not always obtained.

The principle of the method relies on the fact that fine meshes eliminate high frequency errors rapidly but are not effective to remove low frequency errors. To address this issue coarser grids, obtained by removing every other point from the original grid, are used. Therefore, some low frequency errors of the fine grids become high frequency errors for the larger cells of the coarse grids and are thus rapidly removed. By transferring the solution from the fine grids to the coarse grids to compute corrections, and then applying back the corrections to the fine grid, a wider range of error frequencies are removed effectively. The cost of the technique is also relatively low since each level of coarse grid contains 4 times less cells in 2D problems and 8 time less in 3D problems, their computation cost scales by the same factors.

2.2 Unsteady Reynolds-Averaged Navier-Stokes solvers

Unsteady flows are present in many situations, such as turbo-machinery flows, helicopter in flight and aircraft stability studies. As such, time accurate schemes are required to correctly simulate such flows. Solver accelerators described previously, such as multigrid and local time stepping schemes, sacrifice time accuracy to achieve faster convergence. If those accelerators are to be kept, new schemes must be implemented to compute time accurate simulations.

2.2.1 Dual Time-Stepping

Jameson (Jameson, 1991) developed the dual-time stepping scheme in order to benefit from steady state solver acceleration while maintaining time accuracy.

Using a second order backward difference to evaluate the time derivative in equation 2.10

$$\frac{3}{2\Delta t}[W^{n+1}\Omega^{n+1}] - \frac{2}{\Delta t}[W^{n}\Omega^{n}] + \frac{1}{2\Delta t}[W^{n-1}\Omega^{n-1}] + R\left(W^{n+1}\right) = 0 \tag{2.35}$$

where n denote the real time step number.

Equation 2.35 can be treated as a modified steady state problem using timesteps in fictious time t^*

$$\frac{dW}{dt^*} + R^*(W) = 0 \tag{2.36}$$

where the modified residual $R^*(W)$ is defined as

$$R^*(W) = \frac{3}{2\Delta t}W + \frac{1}{\Omega^{n+1}}\left(R(W) - S\left(W^n, W^{n-1}\right)\right) \tag{2.37}$$

with the source term

$$S\left(W^n, W^{n-1}\right) = \frac{2}{\Delta t}\left(W^n\Omega^n\right) - \frac{1}{2\Delta t}\left(W^{n-1}\Omega^{n-1}\right) \tag{2.38}$$

Extending the steady LU-SGS space solver scheme with dual time stepping is straightforward. Using a backward difference to conserve second order time accuracy instead of an Euler difference in 2.11 (Hirsch, 2007), one obtains the following

$$\left(\frac{3\Omega}{2}\frac{I}{\Delta t} + \frac{\partial \bar{R}}{\partial W}\right)\delta W^n = -R\left(W^n\right) \tag{2.39}$$

From there, keeping Δt as the specified step for time integration, the diagonal matrix contribution in equation 2.14 is now

$$\frac{\partial \bar{R}_i}{\partial W_i} = [D] = +\frac{1}{2}|A|_{i+1/2} + \frac{1}{2}|A|_{i-1/2} + A^v_{i+1/2} + A^v_{i-1/2} + \frac{3\Omega}{2}\frac{I}{\Delta t} \tag{2.40}$$

The LU-SGS other operations remain the same as their steady solver counterparts.

2.2.2 Non-Linear Frequency Domain

Many aerodynamic computations aim at simulating unsteady problems which exhibits repeating patterns in time. Such periodic problems are interesting in their periodic steady state, where any variation in the periodic behaviour of the flow is damped out. Since traditional unsteady calculations such as the Dual Time-Stepping method are time accurate, that is the solver integrates the solution in time from an initial state, the solver must compute the transient state of the flow from initial condition to periodic steady state.

The Non-Linear Frequency Domain Method (NLFD) was proposed by Hall *et al.* (2002) and modified by McMullen *et al.* (2001, 2002) to directly solve flow solutions to periodic steady state. Relying on the assumption of periodic flows, Fourier series can be used to represent the solution. Solving the solution as a steady periodic flow in the frequency domain also allows for the use of steady solver's acceleration method, including multigrid and implicit residual smoothing (IRS). The computational cost of the NLFD method thus scales linearly with the number of modes computed with roughly the cost of $(2 \cdot N_{modes} + 1)$ times steady simulation.

Governing equations

From equation 2.10, the state variable vectors and residual vectors are formulated as Fourier series containing N modes

$$W = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{W}_k e^{ikt} \tag{2.41a}$$

$$R(W) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{R}_k e^{ikt} \tag{2.41b}$$

$$\tag{2.41c}$$

To yield

$$\Omega \frac{d}{dt} \left(\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{W}_k e^{ikt} \right) + \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{R}_k e^{ikt} = 0 \tag{2.42}$$

Where $\hat{W}_k$ and $\hat{R}_k$ are the Fourier coefficients of the state vector and residual vector respectively.

A solution of equation 2.42 is

$$ik\Omega\hat{W}_k + \hat{R}_k = 0, \quad k = -\frac{N}{2}, \frac{N}{2} - 1 \tag{2.43}$$

We can thus describe a modified residual in the frequency domain as

$$\hat{R}^*_k = ik\Omega\hat{W}_k + \hat{R}_k = \hat{I}_k \tag{2.44}$$

We then obtain a new equation that we can march in pseudo time τ in order to converge to a solution

$$\Omega\frac{d\hat{W}_k}{d\tau} + \hat{R}^*_k = 0 \tag{2.45}$$

McMullen (McMullen *et al.*, 2001, 2002) showed that convergence acceleration techniques succesfull for equation 2.10 can also be used on 2.45. Using the modified NLFD residual R^* instead of the steady residual R, multi stage Runge-Kutta scheme, implicit residual smoothing and multigrid techniques can be applied to equation 2.45.

2.3 Framework Development

Framework planning is a growing issue in the CFD software development community. Ageing programs becoming harder to maintain because of tool obsolescence, poor programming practices and lack of flexibility. These factors ultimately lead to software performance decrease, longer familiarisation time for new developers trained in recent computational techniques, and increased programming errors (bugs).

For these reasons, many institutions are undergoing the development of brand new frameworks in order to maintain efficiency of currently developed software. For example, *ONERA* launched the elsA project (Thibert) to provide a common platform for all CFD developments of the *ONERA*'s projects. Another example is the American Department of Defence(DoD)'s *CREATE* program (Arevalo *et al.*, 2008) which aims at regrouping the DoD's design tools for ships and aircrafts while making optimal use of high performance computing resources.

The next section discusses in details the approach taken for the present software developments.

CHAPTER 3 SOFTWARE DEVELOPMENTS

3.1 Framework

An important yet often undervalued step when planning any software development project is to choose an appropriate framework. Extensive projects typically involving many programmers and developments spanning over years must be carefully designed. Many framework related problems can undermine the long term usage and updating of a software. For these reasons, extra care was given to the choice of programming practices and used tools in the developed software.

The CFD platform used for the current work is *NSCODE*, a structured mesh based cell-centre RANS and URANS flow solver (Pigeon *et al.*, 2014). The software is currently developed at *Polytechnique Montreal*.

Two programming languages are used to develop *NSCODE*, *C* and *Python*. The *C* language is currently one of the fastest languages apart from machine language for software execution while *Python* is known for its flexibility and easy readability. Computationally expensive parts of the software are thus developed in *C* to obtain optimal performance while high level tasks requiring low computational resources are written using *Python*. The interface between the two languages is handled via the *f2py* library, currently included in the *SciPy* package which includes numerous *Python* libraries to enhance the scientific computing capabilities of the language. Typically, *Python* scripts handle inputs and outputs as well as calling the main modules of the flow solver while the core CFD functions are compiled from *C* source code.

Figure 3.1 shows an example of the execution flow of an *NSCODE* simulation. Each node represents one of the main steps called by the *Python* script.

Since *NSCODE* is used and developed by a growing number of students, the usage of a revision control software is necessary to keep track of the code modifications from multiple developers. Such software allows for easy merging between two versions as it handles file differences and highlights the files affected by multiple changes. Revision control software also ease any debugging process by keeping track of every change made to the source code. Amongst the available revision control software, *Mercurial* was used for *NSCODE* due to its simplicity of use. Using a centralised repository on a server, each developer can upload and download the latest changes in order to keep the whole development team up to date.

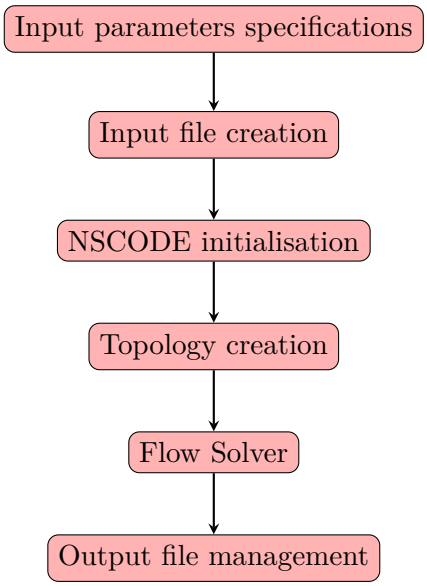


Figure 3.1 Flow chart of steady *NSCODE* execution

3.2 Topology Capabilities Improvements

Flow simulations around complex geometries is one of the challenges of a structured flow solver. Indeed, meshes used by such solvers are Cartesian in the topological space. While such meshes are appropriate to mesh single element airfoils using "O" or "C" type topologies, they are illsuited for geometries containing more than one element such as wing-flap, aileron or spoiler configurations.

To simulate flows around such geometries, a multi block approach is required. The idea is to decompose the domain into different blocks, each block being covered with its own structured mesh. The structured solver can then solve the flow on each blocks. Although each block is solved independently, communication between adjacent blocks must still be assured in order to obtain a valid solution over the whole domain. To allow blocks to communicate the information at their boundaries to adjacent blocks, the implementation of a connection boundary condition is required.

The topology flexibility of the solver is then extended by the implementation of the *Chimera method*. This method allows to treat overset meshes, removing the meshing constraint to have adjacent blocks with matching points at their boundaries (sometimes refered to as one-to-one connections). Overset meshes overlap each other arbitrarily, such that communication through each blocks is assured by interpolations rather than boundary conditions at block edges. This greatly simplifies mesh generation since each geometrical element of a problem can be meshed separately of the others, resulting in higher quality meshes.

3.2.1 Multiblock Solver

The multi-block topology extension contains three main tasks: developing a topology input format to describe blocks topologies, treating the connections between blocks and adapting the solver to treat more than one block.

To enable flow solvers to handle a multiblock mesh, one must develop a way to describe the arrangements of the block relative to one another. In single block meshes, since most of the topologies being of the "O" or "C" type, the topology treatment can be simply hard-coded for these two possibilities. However, for multiblock meshes, we must adopt an efficient input topology file format to describe how the various blocks are connected to each other.

An example of a sample topology input file is showed in Annex A. After a title, the topology input contains the number of blocks and the maximum dimensions of each block in each computational directions. Then, a pattern is repeated to describe each block. The first line of the pattern states the block number and the number of different boundaries it contains. Note that a block can contain more than four boundaries if a boundary does not cover an entire block edge. Then, each boundary condition of the block is described using an additional line. First, the type of boundary is specified using a three character code. Then, the position of the boundary is specified using the starting and ending indices of each computational directions. The last field on each line is used with wall boundary condition to specify which body is associated with the boundary condition. The connection boundary condition takes an additional line to specify which block the boundary is connected to and the associated location of the boundary on the other block.

Once the boundaries can be accurately described with the input file, the software must be updated to handle the connections between the blocks. The algorithm uses phantom cells to enforce boundary conditions. Two layers of phantom cells are used around each block to preserve the stencil which uses third order dissipative scheme throughout the computational domain. The block connection boundary conditions are handled by copying the connecting blocks information into the phantom cells. Using the topology input information, two layers inside the connecting block's domain along the boundary are found and associated with the proper phantom cells. This methodology allows for continuous transfer of information throughout the block's connecting faces, leading to the same solution as an equivalent single block resolution through machine accuracy.

The topology treatment also allows for completely generalised connections between blocks. Blocks can thus connect along any of their faces, including connections along different computational dimensions, that is an **I** normal face connecting with a **J** face, and of opposite

directions, that is when the cells index increase along the boundary in the first block while they decrease along the boundary of second one.

Other than implementing the additional boundary treatment, solver adaptation to multi block meshes is straight forward. Each block can be treated as a separate computational domain and as such each subroutines of the software only need to be nested inside a loop over each block. This method provides an ideal way of using coarse grain parallel computing, separating the computation of each blocks along different CPU's. In the current framework, the shared memory tool *OpenMP* is used for its implementation simplicity although performance is limited to the number of cores sharing the memory.

3.2.2 Overset mesh compatibility

Overset mesh capability, also called chimera method, is the capacity of the solver to handle multiblock meshes were block boundaries do not lie adjacent to one another. Instead, overset grids arbitrarily overlap each other. While this technique greatly reduces meshing constraints on complex geometries, the chimera boundaries between blocks must be handled by the solver.

Typically, a preprocessor is used to treat chimera grids in order to determine which cells will be used to calculate the solution (donor cells), which ones will receive information from other blocks to ensure connectivity (fringe cells) and which ones should be discarded (Hole cells). Also, for each fringe cells, the preprocessor finds the donor cells used to interpolate the solution and their interpolation weight. Some CFD software developers opted to develop standalone chimera preprocessors, such as NASA's *Pegasus* (Suhs *et al.*, 2002). The benefit of standalone preprocessors is the flexibility of working with any flow solver using a standard communication format, such as the CGNS format. The drawback however, is the communication overhead between the chimera preprocessor and the flow solver. While this overhead may be acceptable for steady simulations, it can become an issue when the preprocessing must be done repeatedly such as for unsteady simulations with moving grids. For this reason, the preprocessor, developed within the research laboratory of professor Eric Laurendeau at *Polytechnique Montreal* (Pigeon, 2015; Levesque *et al.*), is included in the *NSCODE* framework. The Chimera preprocessor is added in the workflow after the topology creation step in figure 3.1. A very detailed account of the preprocessing steps can be found in (Pigeon, 2015)

Once the chimera preprocessing is done, a minimal amount of work is required to adapt the flow solver. The first step is to add the chimera interpolation when updating the boundaries. Since all the information required to interpolate the solution on fringe cells is already known, this step consists of looping through the fringe cells and assigning it a state vector consisting

of a weighted sum of its donor cells' state vectors. The second step is to prevent the solution update of fringe and hole cells when updating the solution. This avoids the overwriting of the interpolated values of fringe cells.

Multigrid Issues with Overset Meshes

A major limitation of the chimera method is that convergence to machine accuracy is not achieved using geometric multigrid. The geometric multigrid technique relies on removing one point out of two in all computationnal directions to produce coarser grids. While this technique works well on one-to-one meshes where points on the boundaries are not affected, the algorithm fails when used on chimera grids. Indeed, chimera boundaries can be positioned arbitrarily in the domain. Some point forming the boundary are then removed during the multiblock process. This results in a change in the boundary conditions location from one grid to the other, changing problem definition. Multigrid corrections passed from a level to the other will thus be evaluated for slightly different problems preventing solution convergence to machine accuracy.

This issue can be illustrated using a case of Euler flow around a NACA0012 airfoil. The Mach number is 0.7. The mesh used is the same as the Euler mesh used in section 3.3 cropped at the 65 closest rows to the wall. A cartesian background mesh extending 25 chord lengths in each directions is overlaped to provide far-field resolution.

Figure 3.3 shows the convergence obtained when using two level of multigrid versus no multigrid levels. The simulation using multigrid starts converging slightly faster than the other one which doesn't uses multigrid but floors out after reducing the density residual by about three orders of magnitude.

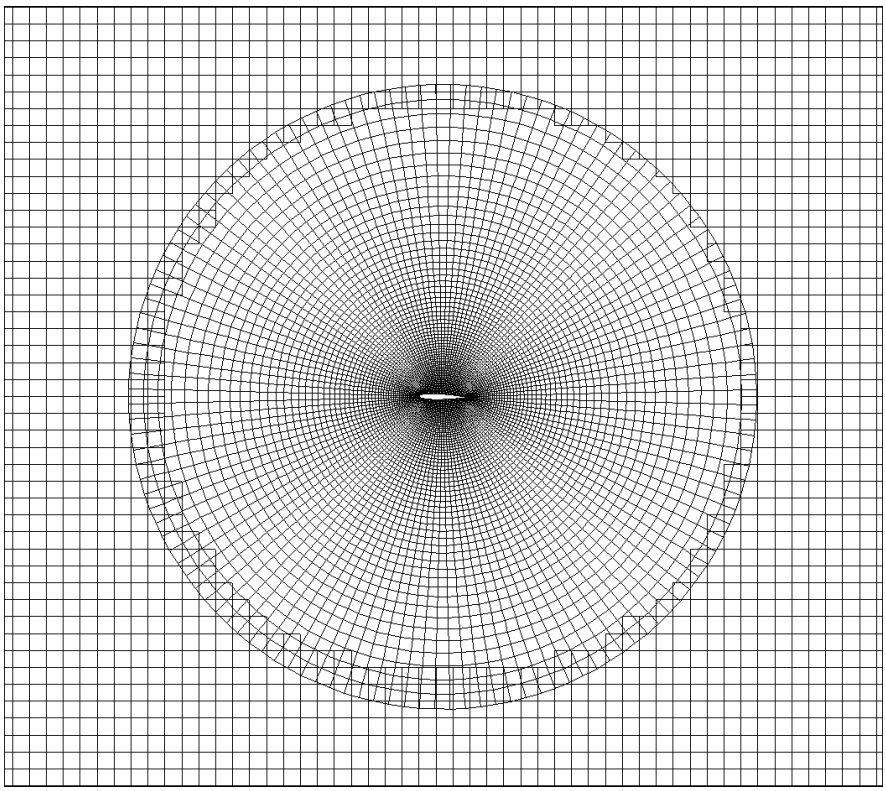


Figure 3.2 NACA0012 overset mesh for testing of multigrid technique on chimera grids

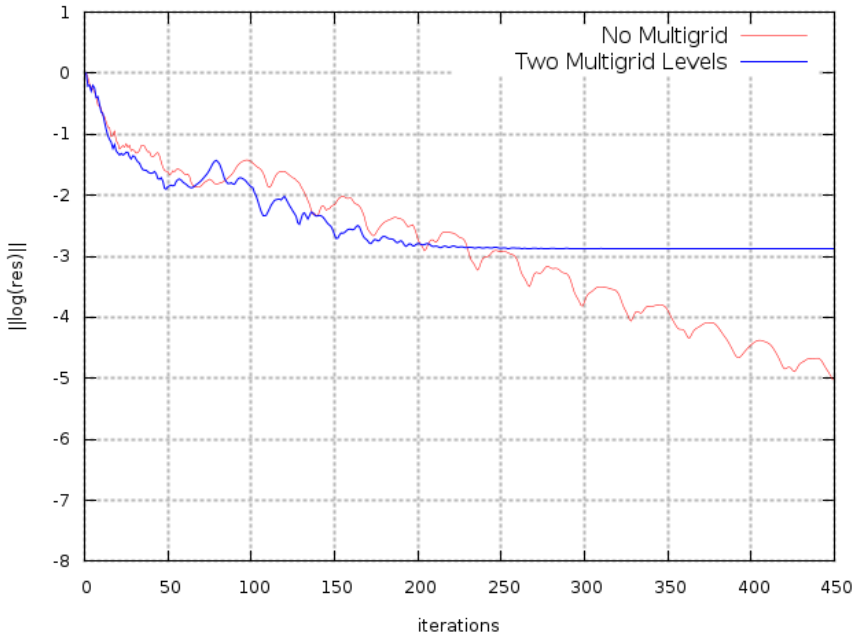


Figure 3.3 Convergence of the NACA0012 airfoil test case with overset mesh

Transonic Biconvex Airfoil

The transonic biconvex airfoil problem is a critical case to test the chimera method since a shockwave is located across the boundary of the two meshes (Soni *et al.*, 2012). Any interpolation errors can thus be amplified. The geometry consist of two NACA0012 airfoils staggered horizontally and vertically by 0.5 chords length. This test case is simulated in an inviscid flow with a free stream Mach number of 0.7 and zero degree angle of attack. For each airfoil, a 128x128 Euler grid generated by Vassberg and Jameson (1981) was used for a total of 32768 cells. No background meshes were used. A buffer of three layer was used with bilinear interpolations.

As can be seen in figure 3.4, the solution yields smooth pressure contours throughout the domain with seamless passage through the chimera boundaries. Figure 3.5 shows the pressure coefficient (C_P) distribution of the upper and lower airfoils compared to Morinishi's (Morinishi, 1992) results with non-body-fitted Cartesian grids.

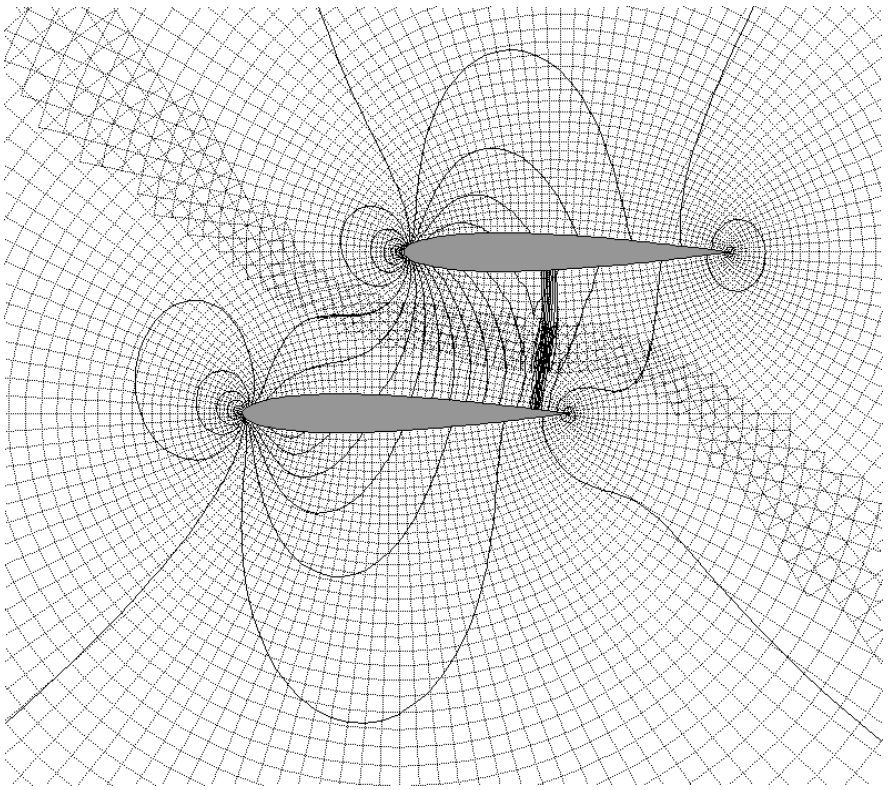
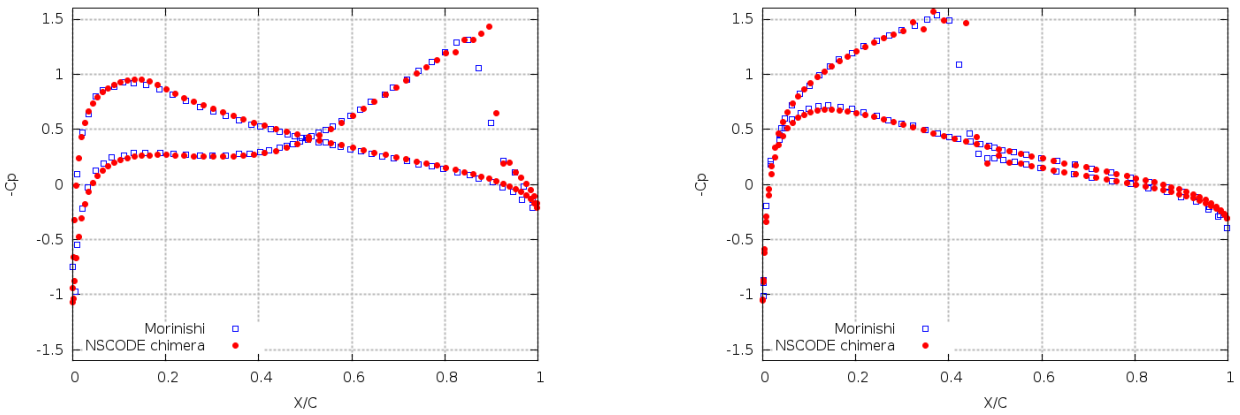


Figure 3.4 Staggered NACA0012 pressure isobars with grid $M = 0.7$



(a) lower airfoil

(b) upper airfoil

Figure 3.5 Staggered NACA0012 pressure coefficients $M = 0.7$

McDonnell Douglas 30P30N Airfoil

The McDonnell Douglas 30P30N Airfoil (MDA) is another interesting test case since it showcases the ability of chimera grids to handle complex geometries with ease. It is also a popular test case in the literature (Liao *et al.*, 2007; Wang and Parthasarathy, 2000; Mavriplis and Mani).

The chimera grids for each element are obtained with a grid generator (NSGRID), developed within the research laboratory of professor Laurendeau at *Polytechnique Montreal*, which uses a blend of parabolic and elliptic approaches (Hasanzadeh *et al.*, 2013). The flap and slat meshes each contain 33000 grid cells whereas the main element mesh contains 132000 grid cells. No background mesh is necessary for this case since the flap and slat meshes are both located inside the main mesh. The first cell is located at 10^{-6} chord off the airfoil surface and an expansion ratio of 1.15 is used to strech the gridlines normal to the wall. Clustering is used at leading and trailing edge regions.

The basic 30P30N MDA is used to compare the solution obtained with chimera meshes to the ones obtained with a one-to-one multi-block mesh containing 256 000 grid cells with similar characteristics. Figure 3.6 shows both meshes. Experimental data is also plotted to compare the C_P distributions. The free stream flow conditions are Mach 0.2, Re 9×10^6 and 16.21˚angle of attack. The Spalart-Allmaras turbulence model is used.

Figure 3.7 shows the convergence obtained while solving the flow. The one-to-one resolutions were made using full approximate storage with 300 iterations on the 3rd grid level, 6000 on the 2nd grid level before solving on the finest grid. Although the 2nd one-to-one grid level shows better convergence rate than the chimera grid due to the usage of multigrid acceleration

technique, the finest level is unable to converge.

Figure 3.7 also shows that the pressure distribution obtained with the chimera implementation of NSCODE matches both the one-to-one resolution of *NSCODE* and the experimental data on the slat and main element. However, the chimera resolution seems to produce an earlier separation at the flap. This behaviour can be explained by our initialization from free stream conditions which tend to produce early stalls on high-lift devices (Rumsey *et al.*, 2011; Deloze and Laurendeau).

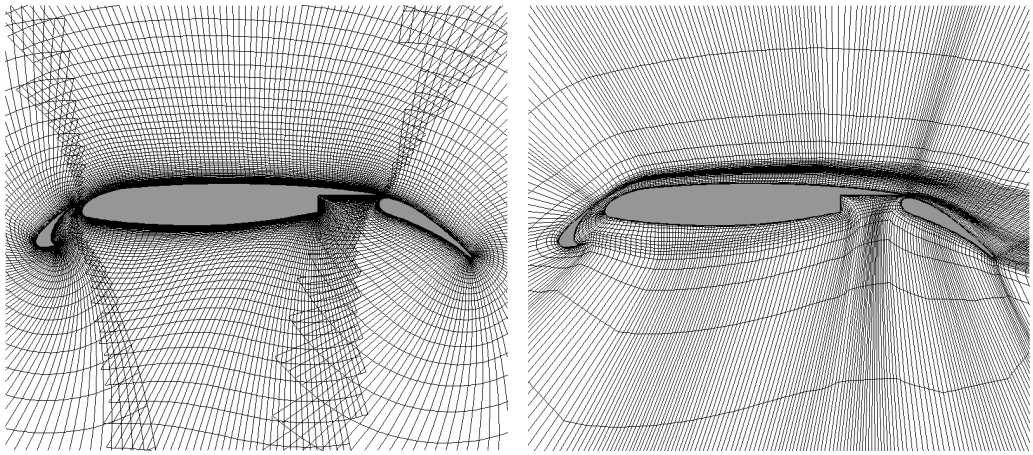


Figure 3.6 Chimera and one-to-one grids used with NSCODE to compute the McDonnell Douglas 30P30N Airfoil test case

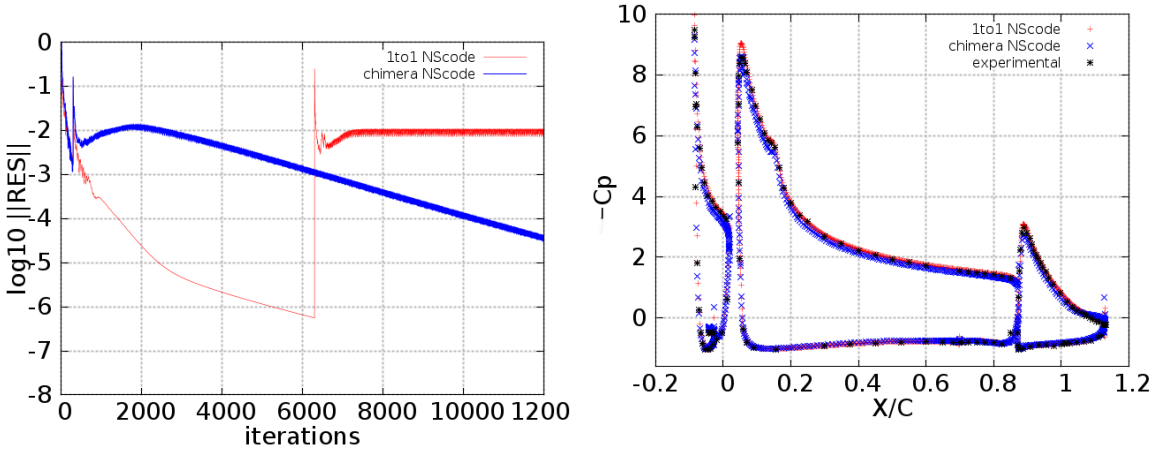


Figure 3.7 Convergence and pressure distribution obtained on the 30P30N airfoil

3.3 Steady Solver Improvements

3.3.1 Matrix dissipation

The matrix artificial dissipation scheme (MATD) implementation is completed using the absolute flux Jacobian described in 2.23. To ensure that the scheme is effective and accurate, the values of V_n and V_l used in equation 2.20 were kept at 0.2 on fine grids and 0.4 on coarse grids. Since The MATD scheme is less dissipative than the scalar dissipation scheme by definition, the artificial dissipation coefficients $\kappa^{(2)}$ and $\kappa^{(4)}$ are increased from $\frac{1}{4}$ and $\frac{1}{128}$ to $\frac{1}{2}$ and $\frac{1}{32}$ respectively.

3.3.2 Point Jacobi

The block implicit Point Jacobi preconditioner was implemented as described in equation 2.28. The method relies on transforming the implicit system of equation 2.11 into a block diagonal, ignoring the off diagonal contributions of the $[U]$ and $[L]$ blocks in equation 2.14. This simplification limits the implicit behaviour within a cell, involving a single four by four matrix inversion for each cell in 2D and five by five inversion in 3D. Since the contributions of other cells to the system is not used, the explicit Runge-Kutta multistep scheme can be used by adding the matricial operator to update the solution as described in equation 2.28. The LU decomposition technique is used to inverse the $[D]$ matrix.

In order to have a good approximation of the value of $\frac{\partial \bar{R}}{\partial W}$ from equation 2.11, Jacobians must be evaluated for the dissipative and viscous fluxes. The absolute flux Jacobian are reused from the matrix-dissipation scheme described above. However, several formulations for the viscous flux Jacobian are used throughout the CFD community. The thin shear layer (TSL) approximation is widely used to simplify the Jacobian formulation. While the TSL approximation is meant to be used normal to the wall boundary, it can be applied in all computational directions (Blazek, 2005). For further simplification, one can also replace the viscous Jacobian with a diagonal matrix containing the viscous spectral radii (Sharov and Nakahashi, 1997). A third formulation of the viscous Jacobian is the one used in NSMB, a 3D structured flow solver developed through European partnerships, as described in Weber (1998). After testing each formulation, it was found that the NSMB Jacobian was the one yielding the best performance and robustness. Each formulation is detailed in Annex B.

3.3.3 LU-SGS scheme

The LU-SGS implicit solving scheme was added to the software in order to achieve higher performance on stretched grids. Many challenges arose and lead to minor changes in the initial scheme in order to achieve desired performance.

As for the Point Jacobi scheme, the absolute flux Jacobian and viscous flux Jacobian must be evaluated at each cell faces. Their formulation in the LU-SGS solving scheme is the same as in the Point Jacobi scheme. In addition, the convective flux Jacobian must be evaluated using the flow properties at each cell centre for the $[U]$ and $[L]$ blocks in equation 2.14, its formulation is based on Pulliam and Steger (1985).

Although it is possible to let $\Delta t \to \infty$ in equation 2.11, additional robustness can be added to the scheme for challenging computations by taking a finite Δt. This fictitious time step's value is determined in the same way as the explicit time step in a purely explicit time stepping scheme. The new term is added to the diagonal of the $[D]$ matrix of equation 2.14. The added diagonal dominance helps stretched meshes to converge properly to a solution at the expense of convergence rate. Setting the CFL number to a very high value will set this term to zero.

In order to compute the LU-SGS domain sweeps, the order of the cells on which equation 2.32 is evaluated is important. On the upward sweep, the cell evaluation order must ensure that the lower cells information is already computed and available while on the downward sweep, the upper cells must be computed first. This problem is treated by sweeping the domain with successive diagonal paths. Figure 3.8 illustrates the diagonal domain sweeping of an upward LU-SGS sweep. The solid dots represent the cells where δW correction is already computed by the current sweep, the empty dot is the cell where δW is currently computed and the other cells are the ones where the results of the previous sweep will be used. The double arrows represent the order in which the domain is covered. In other words, on figure 3.8, the empty dot is the cell where the matrix $[D]$ is inverted, the informations of a solid dot cell will multiply the lower matrix $[L]$ and an empty cell will multiply the upper matrix $[U]$ to find the correction δW of the current sweep on the empty dot cell.

Another important aspect to consider when using the LU-SGS schemes on multiblock meshes is the availability of the information in phantom cells when at the edges of block connections. Without boundary treatment, the off diagonal blocks of the cells alongside the block connections would be lost, reducing the effectiveness of the implicit scheme. To obtain a complete domain implicit treatment, the δW^n obtained after each upward and downward sweeps described by equations 2.32 and 2.34 must be passed through the connecting bound-

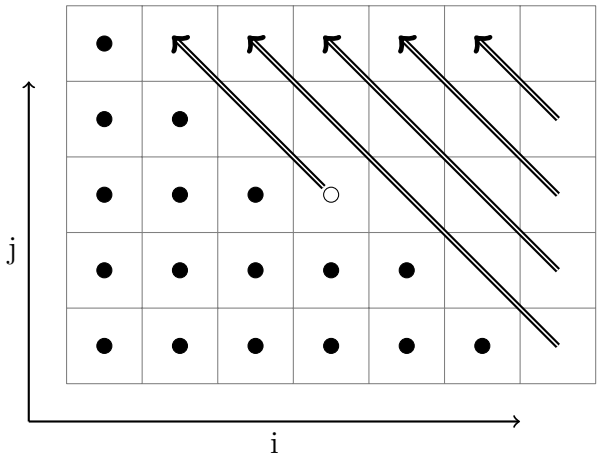


Figure 3.8 Node treatment order of the LU-SGS upward sweeps

ary conditions as described in section 3.2.1.

3.3.4 Validation and Verification

Assessment of the effectiveness of the numerical method was made on numerous test cases. First we validate the matrix artificial dissipation scheme and the space order accuracy of the code. Then, we verify that the new Point Jacobi and LU-SGS solvers converge to machine accuracy without affecting results. In addition, the efficiency of the parallell implementation will be calculated. Also, the LU-SGS scheme will be used with two and four sweeps to compare the performance trade-offs of adding more sweeps in the LU-SGS schemes as described in equation 2.34.

In each test cases, implicit residual smoothing is used for the *Explicit Runge-Kutta* and *Point Jacobi Runge-kutta* methods. In addition, four levels of multigrid are used on each mesh using a W cycle with two fine grids resolutions before and after the cycle.

NACA0012

The first test case chosen is a simple NACA0012 airfoil in an inviscid Euler flow. The 128x128 NACA0012 Euler O-mesh from Vassberg and Jameson (1981) is used. The free stream Mach number is 0.8 and the angle of attack is 1.25˚. The CFL number and LU-SGS relaxation factors(ω) used for each numerical method are listed in table 3.1.

Figure 3.1 shows the convergence rates of the various techniques with respect to multigrid cycles and time. As expected, the LU-SGS method with four sweeps (LU-SGS4) achieved machine accuracy convergence with the lowest number of iterations. The LU-SGS4 scheme

also proved to be the fastest to obtain machine accuracy in term of CPU time, just short of the explicit Runge-Kutta scheme with matrix dissipation. A surprise with this test case is the poor performance of the Point Jacobi preconditioner. It fared no better than the Explicit JST scheme in terms of convergence rate per iterations while taking more CPU work per iteration.

Figure 3.11 show the C_p distribution over the airfoil. As can be expected, oscillations in the region of the shockwave appear when using the MATD scheme. This is due to the less dissipative nature of the scheme compared to the JST scheme which smoothes out these oscillations. In order to assess the effect of the dissipation sheme on these oscillations, the test case is computed using different values of V_n and V_l (0.4, 0.6 and 0.8). Since higher values of V_n and V_l produce absolute jacobian with higher eigenvalues, as dictated by equation 2.20, and thus stronger dissipation, lower oscillations amplitude are obtained.

Table 3.2 lists the aerodynamic coefficients obtained. As expected, the lift and drag coefficients were influenced by the change in artificial dissipation scheme since it changes the right hand side of equation 2.11. The choice of solving scheme however does not influence the results at machine accuracy since they only affect the left hand side of equation 2.11.

Table 3.1 Euler NACA0012 allowable CFL number/ω

Solving scheme	Dissipation scheme	CFL/ω
Explicit Runge-Kutta	JST	7.5
Explicit Runge-Kutta	MATD	7.5
Point Jacobi Runge-Kutta	MATD	7.5
LU-SGS 2 sweeps	MATD	0.6
LU-SGS 4 sweeps	MATD	0.6

Table 3.2 Euler NACA0012 lift and drag coefficients

Solving scheme	Dissipation scheme	C_L	C_{dp}	C_{df}
Explicit Runge-Kutta	JST	0.357869	0.022962	0.000000
Explicit Runge-Kutta	MATD	0.354330	0.022490	0.000000
Point Jacobi Runge-Kutta	MATD	0.354330	0.022490	0.000000
LU-SGS 2 sweeps	MATD	0.354330	0.022490	0.000000
LU-SGS 4 sweeps	MATD	0.354330	0.022490	0.000000

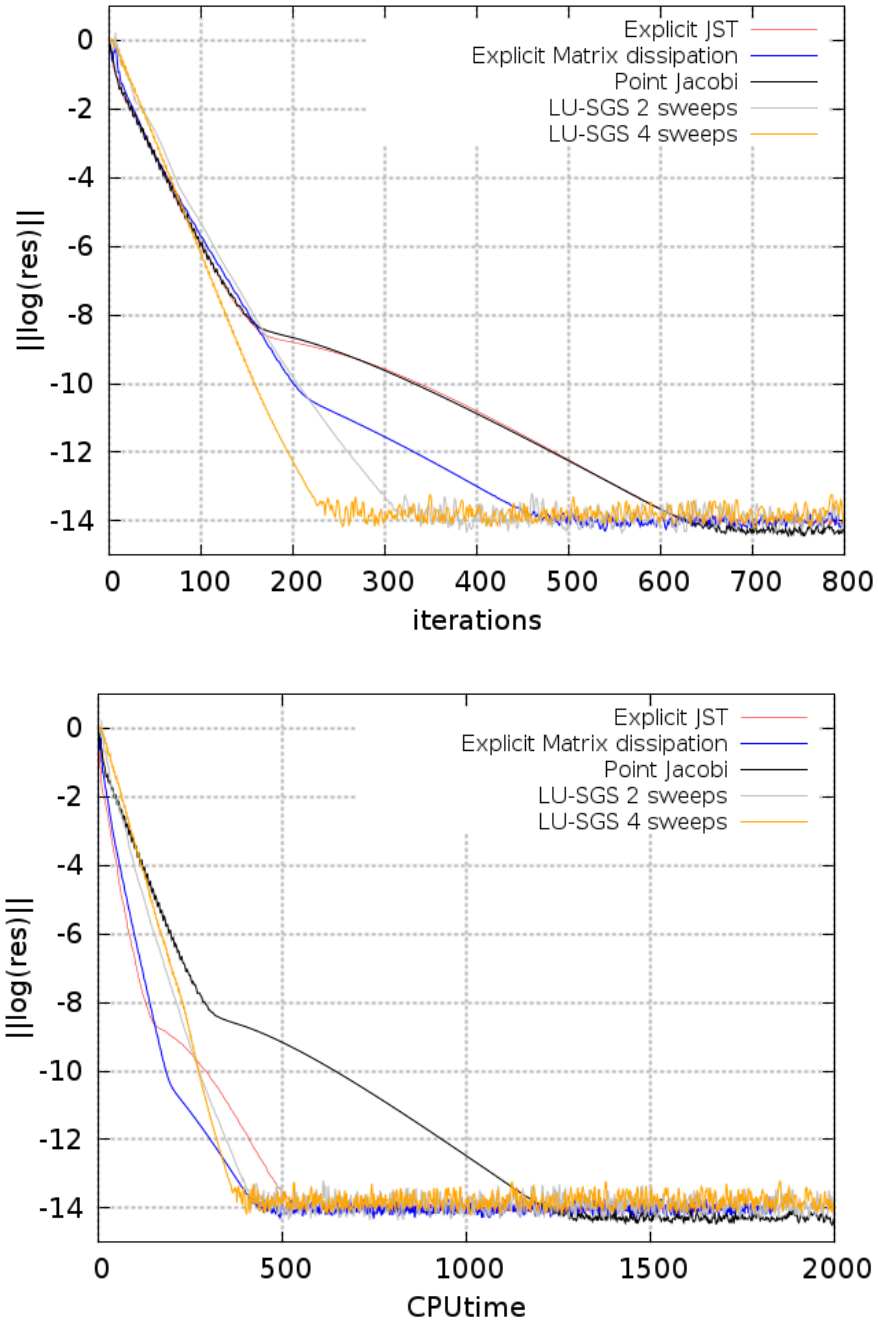


Figure 3.9 Residual convergence of the Euler NACA0012 test case with respect to iterations and CPUtime

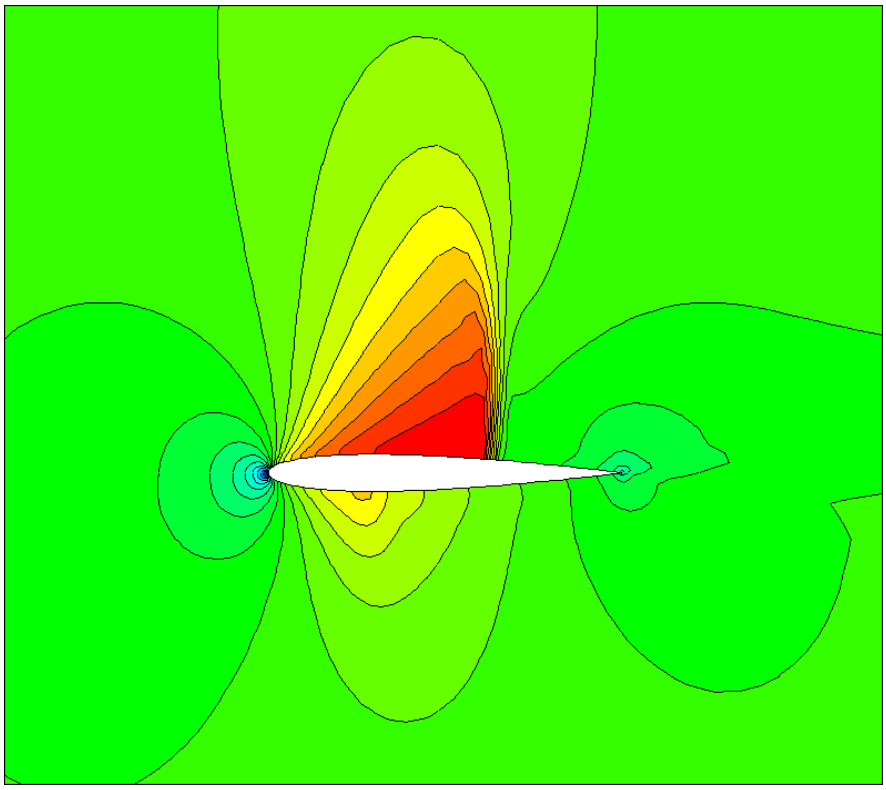


Figure 3.10 Mach contours of the Euler NACA0012 test case

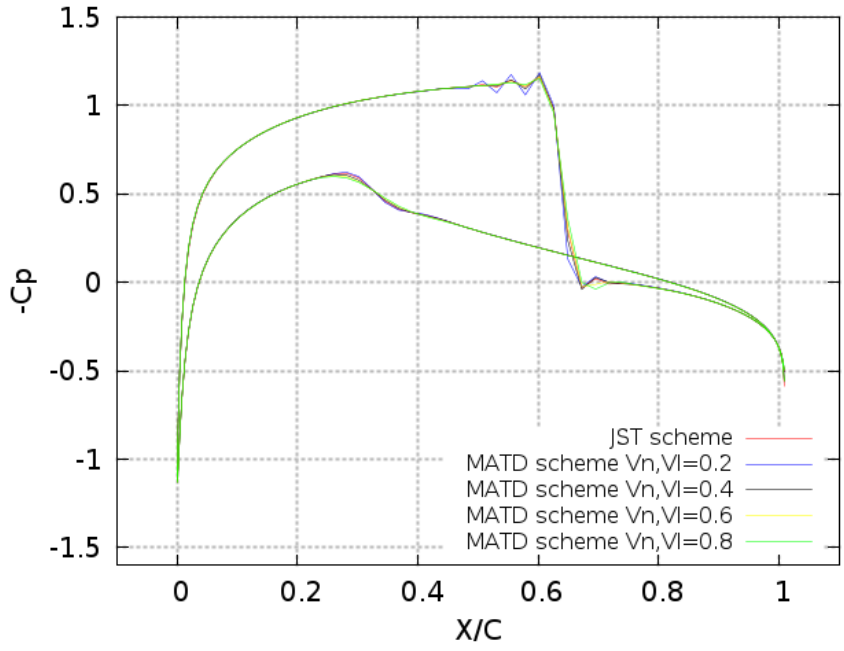


Figure 3.11 Pressure distribution of the Euler NACA0012 test case

This case was also used to assess the parallel computing efficiency of the *OpenMP* implementation. The 128x128 grid used previously is split into 8 blocks of equal size (32x64). Simulation using 1, 2, 4 and 8 processors are launched using both runge-kutta explicit and LU-SGS scheme. The time to complete a multigrid cycle, computed over the average of a hundred cycles, is used to calculate the speedup obtained with the different numbers of processors.

Figure 3.12 shows the speedup curve obtained with *NSCODE*. It is observed that at eight processors, the efficiency of the paralellisation is of 71%. Using Amdahl's law, we obtain a parallelised portion of *NSCODE* of 94.6%.

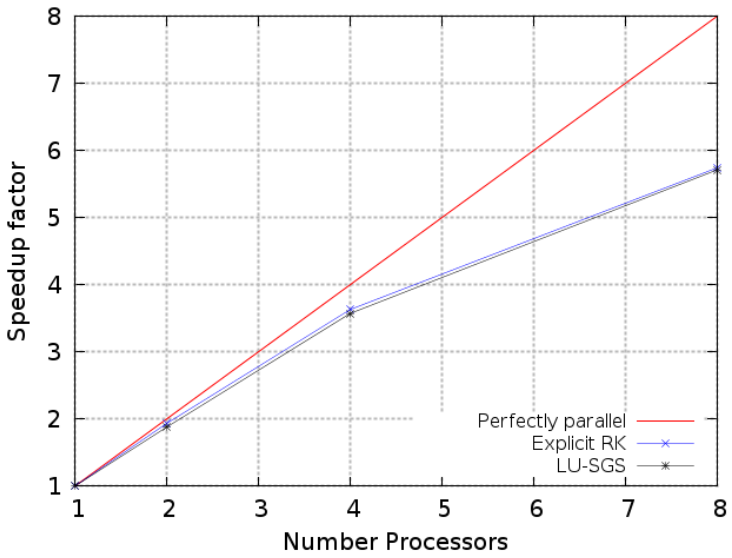


Figure 3.12 Parallel speedup factor of *NSCODE*

Following a similar procedure as Vassberg and Jameson (1981), the Euler NACA0012 test case is used to assess the order of accuracy of the steady solver. In addition to the 128x128 grid used previously, refined grids provided by Vassberg and Jameson (1981) of size 32x32, 64x64, 128x128, 256x256, 512x512, 1024x1024 and 2048x2048 to capture grid convergence of the solution. From these, the order of accuracy p and continuum estimates C_L^* and C_D^* are extracted via the procedure described in Vassberg and Jameson (1981). The freestream flow conditions are unchanged from the test case except for the Mach number of the subsonic study which is decreased to 0.5.

Table 3.3 shows the orders of convergence, the lift and the drag coefficients of established CFD solvers and NSCODE. In the subsonic range, NSCODE is below second order convergence for the lift but shows hyper-convergence for the drag (Pigeon *et al.*, 2014). Although not

shown in the table, a similar hyper-convergence trend was observed by Vassberg and Jameson (1981) with FLO82 for the transonic non lifting case. In the transonic regime, both lift and drag exhibit a first-order-accurate character which is expected due to the artificial dissipation scheme first order switch at shock waves. Continuum estimates for lift and drag coefficients fall within the range of the other CFD methods.

Table 3.3 Convergence order and continuum estimates for subsonic and transonic NACA0012 Euler solutions

	$M = 0.5$				$M = 0.8$			
	$\mathcal{O}(C_L)$	$\mathcal{O}(C_D)$	C_L^*	C_D^*	$\mathcal{O}(C_L)$	$\mathcal{O}(C_D)$	C_L^*	C_D^*
FLO82	2.107	1.805	0.1803	0.000000050	1.151	1.118	0.3562	0.02268
Overflow	1.162	0.820	0.1798	0.000010030	0.438	0.785	0.3517	0.02245
CFL3Dv6	1.153	2.060	0.1804	-0.000000134	0.289	0.960	0.3516	0.02267
NSCODE	1.526	2.654	0.1794	0.000019333	0.928	0.950	0.3529	0.02251

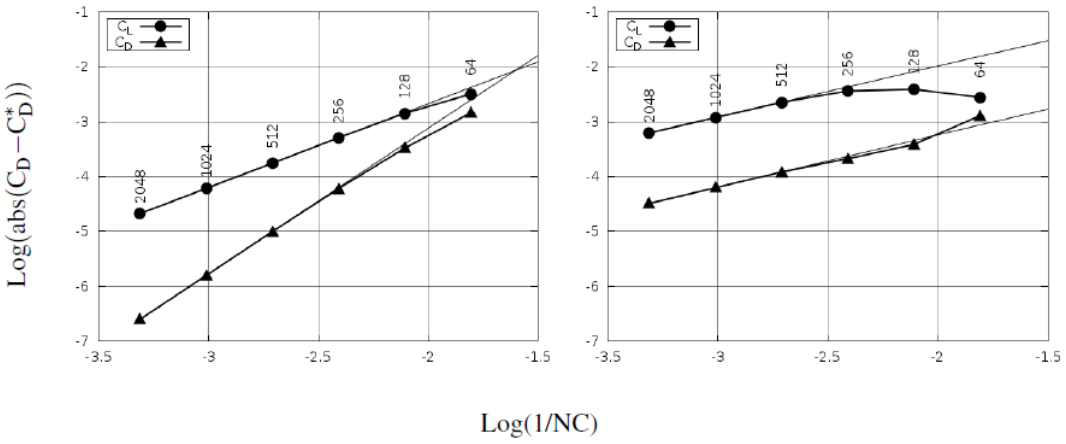


Figure 3.13 Convergence order for subsonic and transonic NACA0012 Euler solutions

RAE2822

The RAE2822 test case is chosen to test the numerical techniques using RANS equations. The free stream Mach number is 0.73 with an angle of attack of 2.79˚and a Reynolds number of 6.5 million. The turbulence model used is Spalart-Allmaras using ten model iterations per multigrid cycle with six ADI sub-iterations as in Cagnone *et al.* (2011). The mesh used contains 18432 cells and the y^+ of the first cells to the wall is inferior to 1. To enhance computing speed, the mesh is split between 6 blocks.

Table 3.4 RAE2822 allowable CFL number/ω of *NSCODE* compared to Cagnone *et al.* (2011)

Solving scheme	Dissipation scheme	CFL/ω	Cagnone *et al.* (2011)
Explicit Runge-Kutta	JST	7.5	-
Explicit Runge-Kutta	MATD	5.0	5.0
Point Jacobi Runge-Kutta	MATD	5.0	7.0
LU-SGS 2 sweeps	MATD	0.4	0.9
LU-SGS 4 sweeps	MATD	0.4	0.9

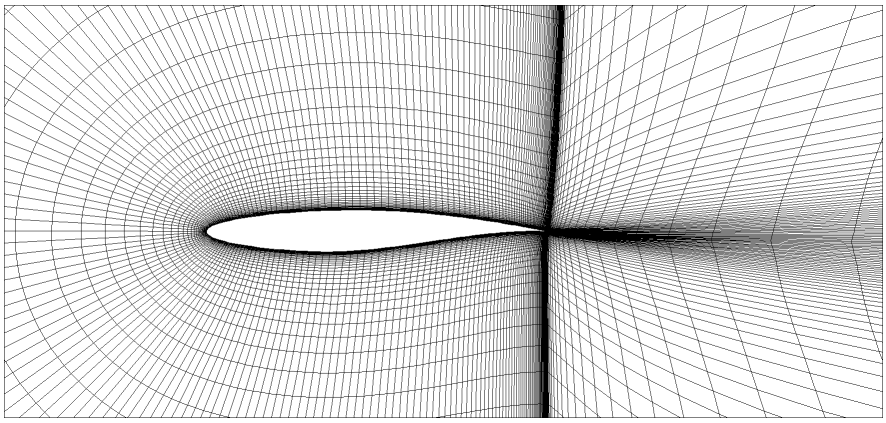


Figure 3.14 RAE2822 grid

One of the difficulties of the test case is the loss of robustness due to the less dissipative MATD scheme. Since the mesh used contains stretched cells in the boundary layer and wake regions, the reduced robustness of the MATD scheme forced to reduce the CFL number from 7.5 to 5.0 for simulations using the scheme as summarised in table 3.4. This explains the good performance of the explicit JST scheme in convergence rate versus time seen in figure 3.15. Another observation is that the Point Jacobi preconditioner which seemed ineffective for the previous test case now yields the best convergence rate for the number of iterations. It seems that the point implicit preconditioning is beneficial when used on stretched cells. Note that this behavior has also been observed by Cagnone *et al.* (2011).

Table 3.5 RAE2822 lift and drag coefficients

Solving scheme	Dissipation scheme	C_L	C_{dp}	C_{df}
Explicit Runge-Kutta	JST	0.755749	0.010302	0.005974
Explicit Runge-Kutta	MATD	0.755546	0.010340	0.005944
Point Jacobi Runge-Kutta	MATD	0.755546	0.010340	0.005944
LU-SGS 2 sweeps	MATD	0.755546	0.010340	0.005944
LU-SGS 4 sweeps	MATD	0.755546	0.010340	0.005944

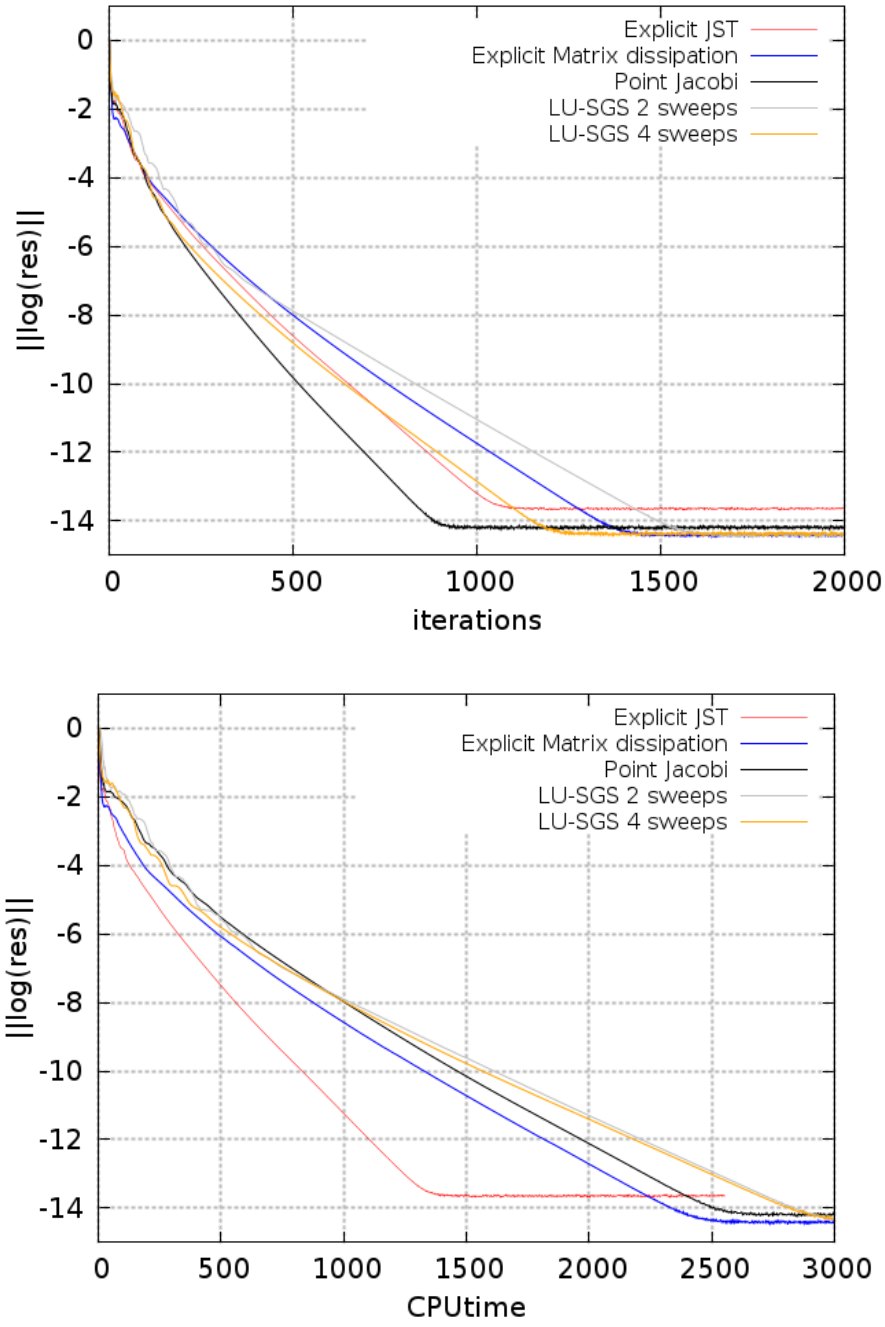


Figure 3.15 Residual convergence of the RAE2822 test case with respect to iterations and CPUtime

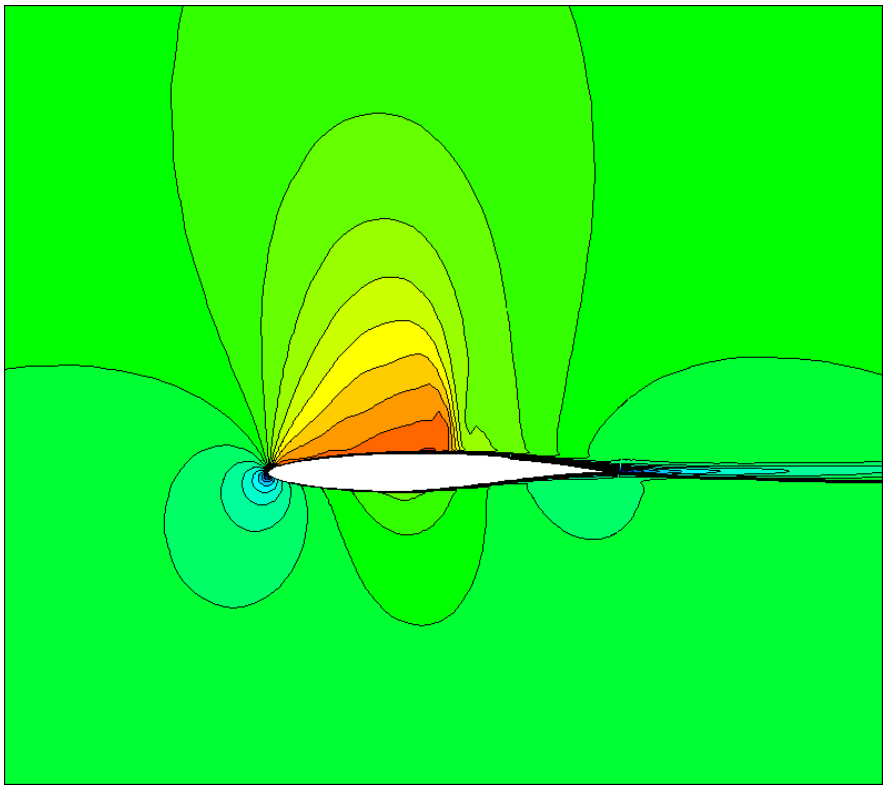


Figure 3.16 Mach contours of the RAE2822 test case

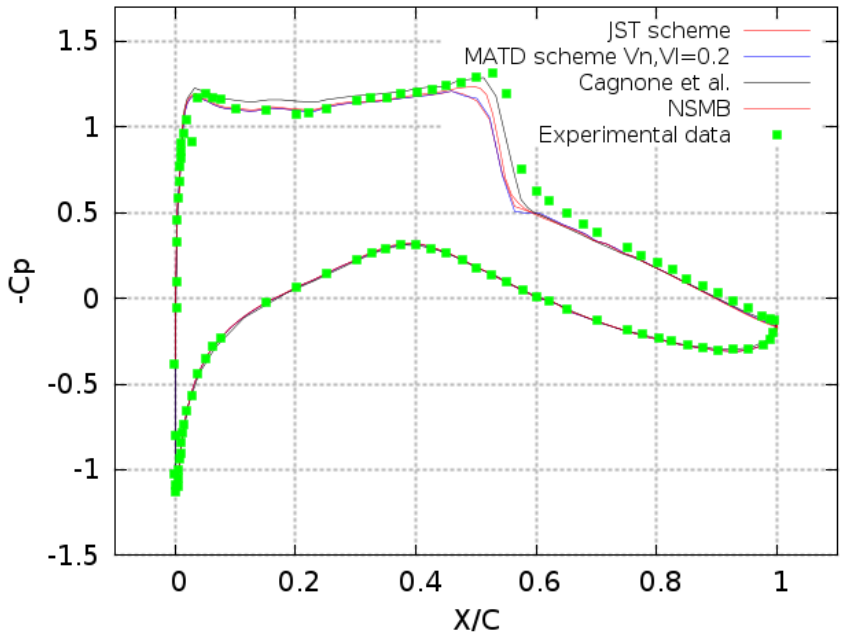


Figure 3.17 Pressure distribution of the RAE2822 test case

NLR7301

The objective of the NLR7301 test case is to asses the performance of each solver over a two element airfoil. The free stream Mach number is 0.185 with an angle of attack of 13.1˚and a Reynolds number of 2.51 million. As with the previous test case, the turbulence model used is Spalart-Allmaras using ten model iterations per multigrid cycle with six ADI sub-iterations. The mesh, provided by *Bombardier Aerospace* (Cagnone *et al.*, 2011), contains 145k cells between 9 blocks and the y^+ of the first cells to the wall is about 0.5. Since the angle of attack is relatively high, computing stability is improved by gradually increasing (ramping) the simulation angle of attack from 3.0˚to the final value of 13.1˚using 400 multigrid cycles, as done in Cagnone *et al.* (2011).

Table 3.6 NLR7301 allowable CFL number/ω of *NSCODE* compared to Cagnone *et al.* (2011)

Solving scheme	Dissipation scheme	CFL/ω	Cagnone *et al.* (2011)
Explicit Runge-Kutta	JST	7.5	-
Explicit Runge-Kutta	MATD	5.0	2.0
Point Jacobi Runge-Kutta	MATD	5.0	2.0
LU-SGS 2 sweeps	MATD	0.2	0.9
LU-SGS 4 sweeps	MATD	0.2	0.9

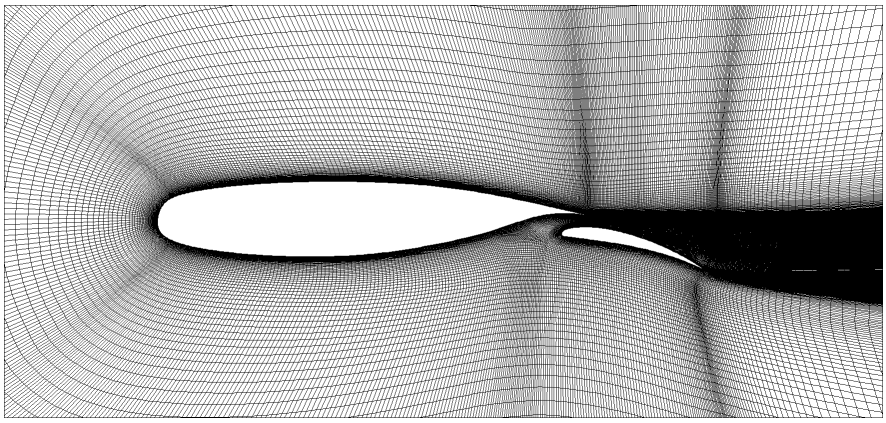


Figure 3.18 NLR7301 grid

The results shown are taken after 2000 iterations of multigrid cycles. The first surprise with this test case is that while the Explicit Runge-Kutta scheme showed convergence rate, the Point Jacobi preconditionner and the LU-SGS schemes picked up unsteady modes and had their convergence stalled as illustrated in figure 3.20. This observation is contrary to what Cagnone *et al.* (2011) observed on the same test case. Since convergence of the residual to

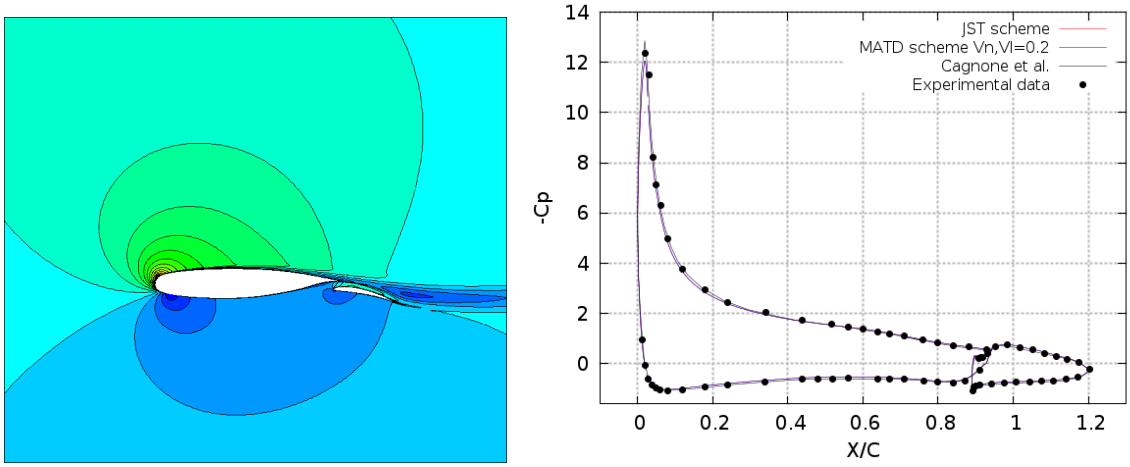


Figure 3.19 Mach contours and pressure distribution of the NLR7301 test case at 13.1˚ angle of attack

machine accuracy is not reached, the values of the aerodynamic coefficients differ from one solver to the other in table 3.7. When simulating the same test case at 5˚angle-of-attack however, the flow remains completely attached and all solvers show good convergence as can be seen in figure 3.21. Table 3.8 also show accordance on the aerodynamic coefficients between the different solvers used.

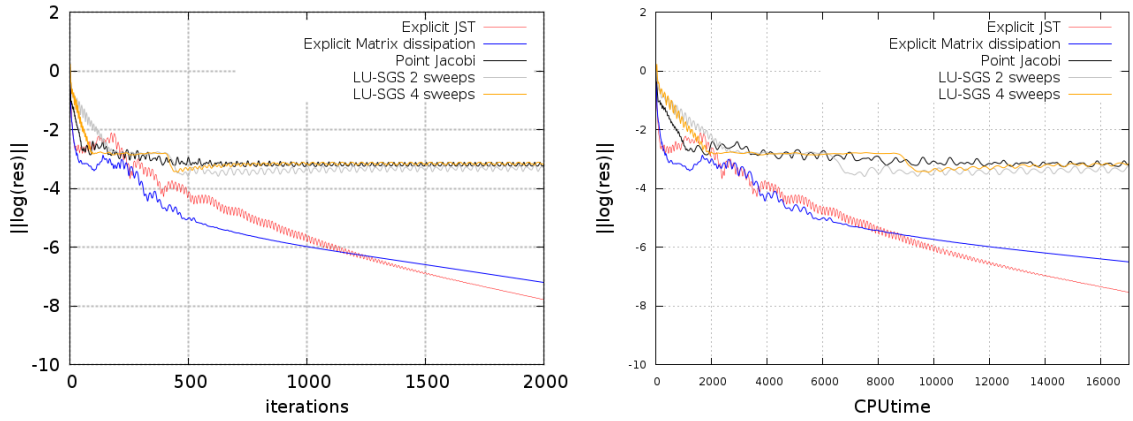


Figure 3.20 Residual convergence of the NLR7301 test case with respect to iterations and CPUtime at 13.1˚ angle of attack

Table 3.7 NLR7301 lift and drag coefficients at 13.1˚ angle of attack

Solving scheme	Dissipation scheme	C_L	C_{dp}	C_{df}
Explicit Runge-Kutta	JST	3.098091	0.088230	0.008983
Explicit Runge-Kutta	MATD	3.096838	0.088610	0.009001
Point Jacobi Runge-Kutta	MATD	3.105250	0.088540	0.009003
LU-SGS 2 sweeps	MATD	3.109561	0.092050	0.009002
LU-SGS 4 sweeps	MATD	3.106544	0.085808	0.009020

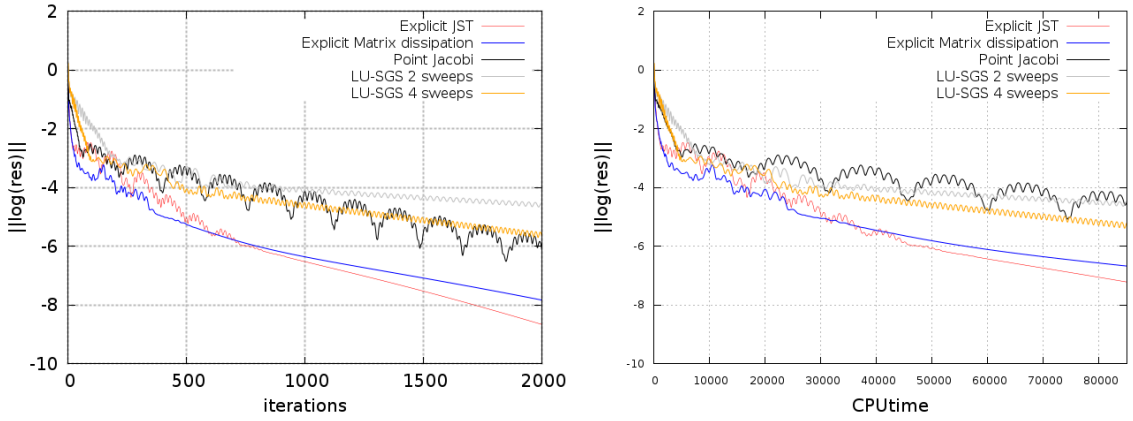


Figure 3.21 Residual convergence of the NLR7301 test case with respect to iterations and CPUtime at 5˚ angle of attack

Table 3.8 NLR7301 lift and drag coefficients at 5˚ angle of attack

Solving scheme	Dissipation scheme	C_L	C_{dp}	C_{df}
Explicit Runge-Kutta	JST	2.225654	0.037163	0.010748
Explicit Runge-Kutta	MATD	2.224494	0.037486	0.010749
Point Jacobi Runge-Kutta	MATD	2.224494	0.037486	0.010749
LU-SGS 2 sweeps	MATD	2.224494	0.037486	0.010749
LU-SGS 4 sweeps	MATD	2.224494	0.037486	0.010749

3.4 Unsteady solvers

3.4.1 Dual Time-Stepping

The addition of the *Dual Time Stepping scheme* (DTS) to the framework required minor changes to the software. In addition to a simple source term addition, the implementation took advantage of the two loops over pseudo time and real time. The loop over pseudo time is very similar to the steady flow solver's pseudo time marching. As such, the steady flow solver was modified to calculate the additional terms in the residual described in equation 2.37.

Since the loop over real time requires several user inputs and little computation costs, it is placed inside the python script. This choice easily gives users the flexibility to launch complex simulations. Indeed, it is often observed that initialising an unsteady problem by using a few multigrid cycles of steady flow solvers can reduce the simulation time put on the transient evolution of the flow. This methodology also allows to start an unsteady simulation with a large timestep to quickly observe periodic steady state and then change to a finer timestep to study the flow with good time accuracy. Figure 3.22 is an example of flowchart for an unsteady simulation using the steady solver for initialisation.

3.4.2 Non Linear Frequency Domain Solver

As described in section 2.2.2, the Non Linear Frequency Domain method is a very efficient method for simulating periodic flows. However, the implementation of such a solver requires extensive software architecture adaptation. In order to obtain the Fourier coefficients described in equation 2.41, several solutions along the time period are needed depending on the number of modes in the Fourier series. The flow solver must compute $2N + 1$ solutions in parallel at different points in time, or time samples, in order to complete the Fourier transforms, N being the number of modes. Time thus effectively becomes an additional dimension of the computational domain. It is important to note that computational costs in CPU time and memory requirements also scale linearly with the number of time samples.

The figure 3.23 represent the mains steps of the NLFD solver. From the set of Fourier coefficient of the conservative variables $\hat{W}$, an inverse Fourier transform is used to get the conservative variables W at each time sample. Then, the residual as defined in equation 2.9 is calculated from the state vector at each time sample. A Fourier transform is then used to obtain the Fourier coefficient of the residuals $\hat{R}$. From the Fourier coefficients of the residual and of the state vector, the modified residual $\hat{R}^*$ can be computed. Iterations are then made

by updating the Fourier coefficients of the solution $\hat{W}$ using $\hat{R}^*$. Since every time sample must be computed in paralell in order to find the Fourier coefficients, W, R, $\hat{W}$ and $\hat{R}$ are bidimensionnal matrices of size $2N+1$ by the number of conservative variables within each cell. The dimension of size $2N+1$ contains the solution at each time sample for W and R or the Fourier coefficients for $\hat{W}$ and $\hat{R}$.

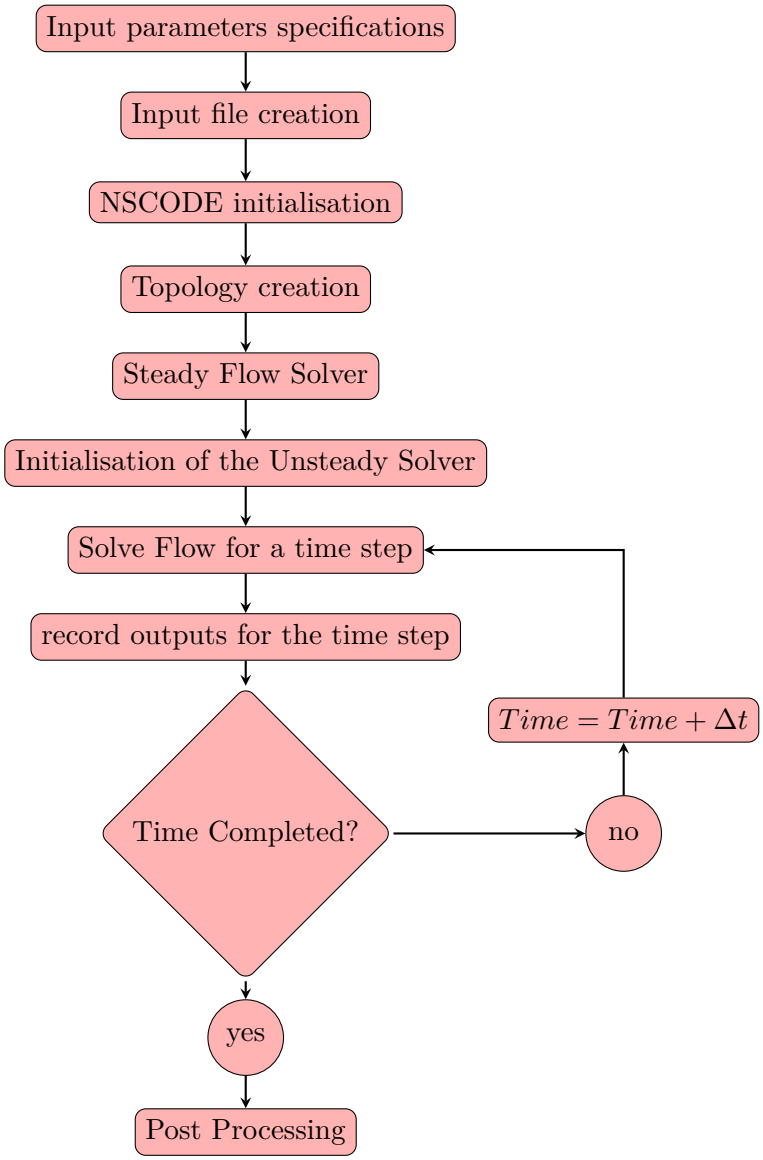


Figure 3.22 Flow chart of unsteady *NSCODE* Dual Time Stepping execution

To initialise NLFD simulations involving moving geometries, the same procedure as for the steady solver is applied to each time sample, which is giving the far field boundary values to each cells. The moving meshes will provide different solutions at each time samples and the higher modes of the solution will appear naturally. On non-forced unsteady simulations

however, as the geometry is not moving, higher modes may not appear since each time samples are identical. Some strategies are thus required to obtain convergence of the solution on static geometries. For the purpose of this work, the strategy used is to impose a movement (pitching or plundging) to the geometry for a few multigrid cycles before starting the computation of the static case. This allows all the different modes to appear as the simulation starts.

In the current framework, the FFTW (Fastest Fourier Transform in the West) library (Frigo and Johnson, 1998) is used to compute the direct and inverse Fourier transforms. The library, developed at the *Massachusetts Institute of Technology*, is optimised to compute Fourier transforms with minimal computing costs. Using a third party optimised library for this step thus saves development and simulation time.

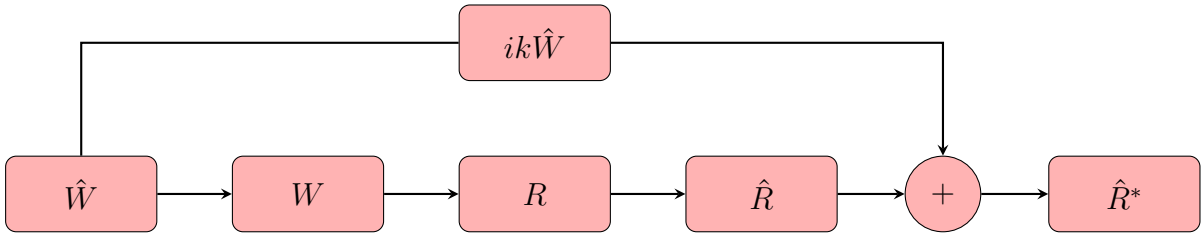


Figure 3.23 Flow chart of NLFD iteration algorithm on the state variables

3.4.3 Arbitrary Lagrangian Eulerian formulation

Many unsteady flow problems involve moving geometries. Using rigid grid movement, a model must be used to account for the motion of the grid relative to the fluid. The principle of the *Arbitrary Lagrangian Eulerian* (ALE) formulation is to add to the convective fluxes the part that is due to the sweeping of the cell faces through the fluid. Since this work only covers test cases with rigid grid movement, the Geometric Conservation Law (GCL), allowing arbitrary grid movement, is not verified. Modifying the formulation of the convective flux in equation 2.3

$$\vec{F_C^M} = \vec{F_C} - V_t\vec{W} \tag{3.1}$$

Where V_t is the mesh contra-variant velocity, in 2D

$$V_t = n_x\frac{\partial x}{\partial t} + n_y\frac{\partial y}{\partial t} \tag{3.2}$$

In practice, this formulation means that the difference between the flow velocity and mesh velocity must be used to compute the convective flux vector. Along with this change, one must also make sure that the rest of the numerical schemes remains consistent with the

modification of the flux vector. The velocities used in convective and absolute flux Jacobian and their eigenvalues must thus also use the difference between the fluid and mesh velocities. The treatment of the boundary conditions make no exceptions and must also use this velocity difference.

3.4.4 Validation and Verification

Pitching NACA0012

The first test case is a pitching NACA0012 airfoil under Euler conditions. The freestream Mach number is 0.755. The sinusoidal pitching oscillation mean angle of attack (α_0) is 0.016˚ and the oscillation amplitude (α_A) is 2.51˚with a reduced frequency (k) of 0.0814. The center of rotation is located at the quarter chord point of the airfoil. The mesh is reused from the previous NACA0012 Euler test case from section 3.3.4. For the dual time-stepping solver, the mesh displacement and velocity is calculated analytically at each time step. For the NLFD function, the mesh velocity and position are calculated analytically at each time sample and stored as such.

The dual time-stepping simulation was made using 150 multigrid cycles of steady simulation followed by 2 period of coarse time resolution to compute the transients without spending too much time. The final time resolution was made using 100 time steps per period. The density residual is reduced by ten orders of magnitude at each time step.

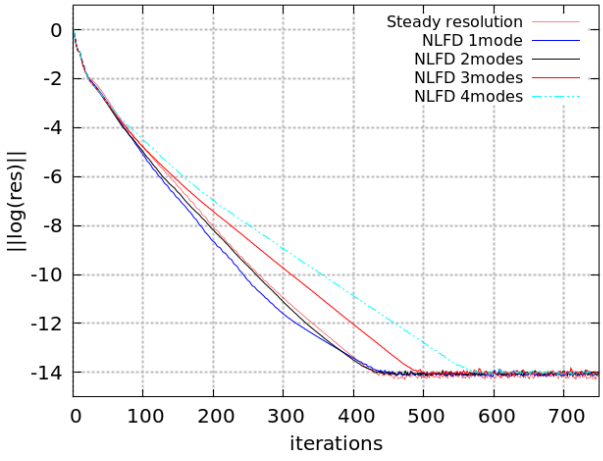


Figure 3.24 Euler pitching NACA0012 convergence characteristics versus multigrid cycles for different number of modes

The NLFD simulation was made using 1 through 4 modes. This test case will be used to verify the claims of the NLFD method to possess the same convergence rate in terms of iterations as

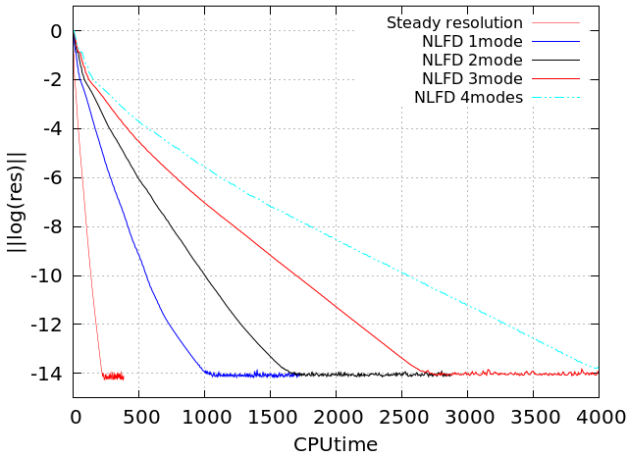


Figure 3.25 Euler pitching NACA0012 convergence characteristics versus CPU time for different number of modes

a steady resolution. It will also verify that the computing cost of the solutions grows linearly with a multiple of $2N+1$. Table 3.9 shows the time to converge to machine accuracy, extracted from figure 3.25, for each number of modes used for the NLFD simulations. From figure 3.24, it is concluded that convergence to machine level is obtained after about 500 iterations. As can be observed, the NLFD computational cost scales as expected when varying the number of modes of the Fourier series. The CPU time for an equivalent steady computation with the same flow solver settings (CFL number, multi-grid levels, dissipation scheme) is added as reference. The *objective CPU time* column represents the time each NLFD simulation is expected to take if the direct and inverse Fourier transform are performed instantanously. It is obtained by multiplying the steady resolution time by the factor of $2N+1$. The difference between the actual CPU time and the Theorical CPU time show that the Fourier transforms take about 33% of the total CPU time.

Table 3.9 CPU time to achieve 500 multigrid cycles for different number of modes used in the NLFD resolution for the Euler pitching NACA0012 case

Number of modes	CPU time(s)	Objective CPU time(s)	FFT weight in simulation time
Steady resolution	257	-	-
1mode	1149	771	32.9%
2modes	1917	1285	33.0%
3modes	2725	1799	34.0%
4modes	3580	2313	35.4%

This test case is also used to test the chimera capabilities of the software with unsteady simulations. Simulations of the test case using an overset mesh are completed with both

technique, DTS and NLFD method. This test case is ideal to test the chimera capability as it uses moving grids, the chimera preprocessor is thus called several times throughout the resolution. This means the solver must be able to communicate with the preprocessor to update the chimera interpolation informations for each time step or time sample depending on the time discretisation method used. The overset mesh used is the same as the one used previously in section 3.2.2 and illustraded in figure 3.2. The overset mesh simulation is done using two modes for the NLFD resolution.

Figure 3.26 shows the lift and drag coefficient hysteresis versus the instantaneous angle of attack of the airfoil. Good agreement between the time accurate DTS solver, the NLFD solver, the *PMB* software solution(Da Ronch *et al.*, 2013) and experimental data (Landon). The chimera results also show very good agreement with the other results obtained using the one-to-one grid. The NLFD method yields good results when using at least two modes for the resolution. Indeed, since the evolution of the drag coefficient during the period has two maxima and minima, a single mode is not sufficient to describe such a shape, which would lead to inconsistent drag prediction.

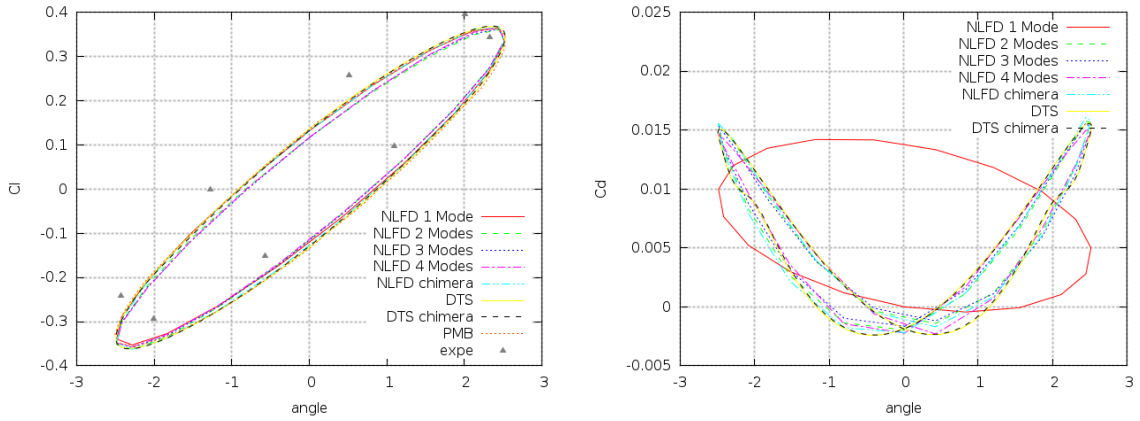


Figure 3.26 Lift and drag hysteresis of the lift and drag coefficients versus the instantanous angle of attack

The time accuracy order of the DTS scheme in NSCODE is also assessed using this test case. The integral of the L_2 norm of C_L and C_D are recorded in the steady periodic flow state to assess the accuracy orders. These parameters are computed as follows

$$\|C_L\| = \sum_{i=1}^{n} C_L(t_i)^2 \Delta t \tag{3.3a}$$

$$\|C_D\| = \sum_{i=1}^{n} C_D(t_i)^2 \Delta t \tag{3.3b}$$

Where n is the number of time steps over a period(T), Δt is the time step and $C_L(t_i)$ and $C_D(t_i)$ are respectively the value of the lift and drag coefficient obtained at the timestep i. The test case is computed using different values of time steps varying from 10 time steps per period up to 160 time steps per period.

As can be observed in table 3.10, both lift and drag norms achieve second order convergence with respect to time resolution. This is expected as a second order backward difference scheme is used in the solver to compute the time derivatives. The convergence of the error on the L_2 norms of each coefficient is illustrated in figure 3.27.

Table 3.10 Convergence order and continuum estimates for pitching NACA0012 Dual Time-Stepping solutions

Time steps per period	$(\Delta t/T)$	$\|C_L\|$	$\|C_D\|$
10	0.1	0.2472891128	0.009104167
20	0.05	0.2550601978	0.0093415096
40	0.025	0.257244761	0.0093999994
80	0.0125	0.2577983417	0.0094117931
160	0.00625	0.2579377802	0.009414387
320	0.003125	0,257972786	0,009414983
Continuum		0,257984521	0,009415161
Order p		1,993945954	2,121784556

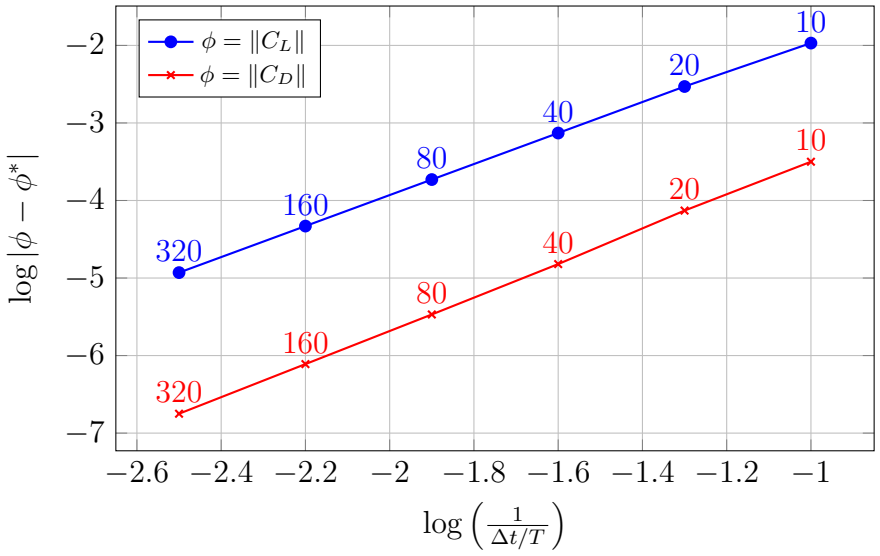


Figure 3.27 Order of time convergence of *NSCODE* using the L_2 norm integral of lift and drag coefficient over a period for a pitching NACA0012 airfoil

Cylinder

The second test case is the observation of Von Karman vortices in the wake of a stationary circular cylinder in laminar flow. The free stream Mach number is 0.3 and the Reynolds number is 100. The objective of this test case is to show good agreement between the dual time-stepping, the NLFD method and the literature. Figure 3.28 shows the mesh used for the test case, which is a "O" mesh geometry containing 10000 cells with a wall grid spacing of 10^{-4} chord obtained from NASA's CFL3D website[1].

Using the DTS solver, the transient part of the flow is accelerated using asymmetric initialisation. This is done by computing the problem with the steady solver at a 45° angle-of-attack and then switching back to a zero angle of attack for unsteady resolution. Without an asymmetric initialisation, the symmetric grid would lead to symmetric vortices, which are known to be non-physical, until machine errors would introduce small asymmetries in the flow that would grow into vortex shedding. For each time step, the solution is advanced in pseudo time to decrease the density residual by five orders of magnitude. To compute the transient part between initialisation and steady periodic flow, three unsteady loops have been used with increasing time resolution. Each initialisation loop lasts four periods, and contain respectively 5, 20 and 40 timesteps per period. The period used is based on the Strouhal number found in the literature (Mosahebi and Nadarajah, 2013). The final resolution loop for the periodic flow contains 100 points per period. The explicit Runge-Kutta space resolution results are compared to the ones obtained with the LU-SGS scheme.

Table 3.11 Strouhal numbers obtained for the cylinder under laminar flow conditions

	St
Mosahebi and Nadarajah (2013)	0.16704
Explicit RK DTS	0.17458
LU-SGS DTS	0.17458
Experimental results (Williamson, 1989)	0.16434

Using the NLFD method, the solution was initialized with 100 multigrid cycles where the cylinder oscillates in a plunging motion. Without this process, all time samples would have identical convergence and only the 0th mode would be non zero, one would thus need to wait for machine errors to induce some non-zero values in the other modes of the solution.

Since the NLFD method requires to know in advance the exact Strouhal number in order to achieve machine accuracy, the Strouhal number obtained with the DTS simulation is used to compute the NLFD simulation. However, the Strouhal number obtained from the DTS

[1] `http://cfl3d.larc.nasa.goc/Cfl3dv6/cfl3dv6_testcases.html`

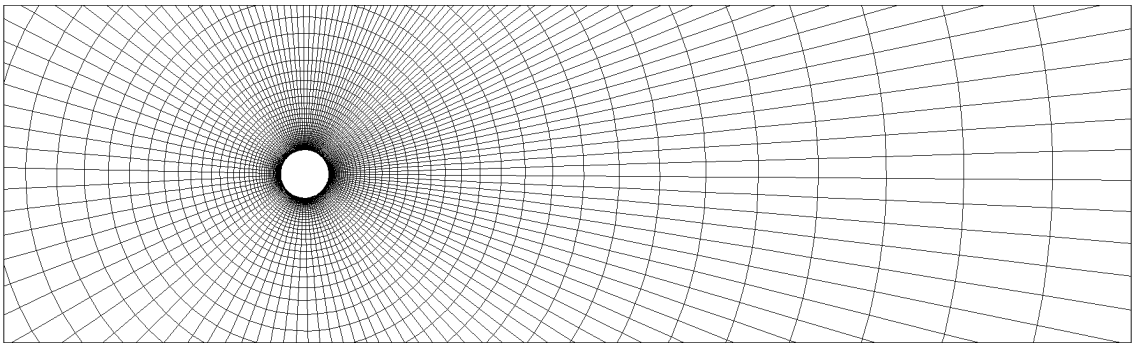


Figure 3.28 Mesh used for the circular cylinder simulations

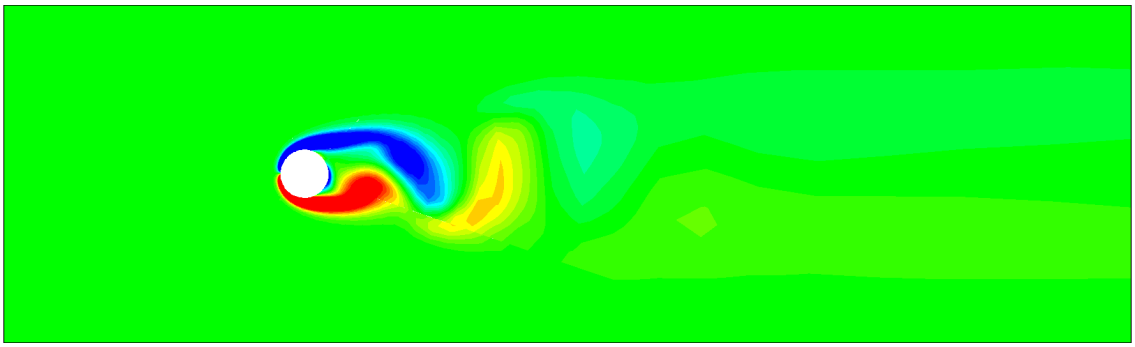


Figure 3.29 Vorticity contours on the circular cylinder simulations

solver may not perfectly match the NLFD resolution. Analogously to a steady solution that doesn't reach convergence on an unsteady test case, a wrong Strouhal number won't allow the unsteady terms to balance perfectly the convective and viscous terms of the residual. Therefore, density residual convergence only reached four orders of magnitudes. Table 3.11 compares the Strouhal numbers from the literature, experimental results and NSCODE's dual time stepping. The simulation was made using 4 modes for the Fourier series.

Multigrid technique was not used for this test case neither for the dual time stepping nor for the NLFD simulations. The reason being that the bigger cells and the use of first order artificial dissipation on coarse meshes lead to steady solutions of the flow on coarse grids, changing the physics of the problem. The change in physics between fine and coarse grids lead the multigrid technique to transfer wrong corrections to the fine grid, altering the solution. This situation is comparable to supersonic flows where multigrid tends to affect the solution.

Figure 3.30 shows the agreement between the implemented unsteady solvers. Each simulation yielded identical curves for lift and drag coefficients variation along a period. Figure 3.29 shows the vorticity field behind a circular cylinder. It is observed that the vortices dissipated after approximately four chords lengths. This is mainly due to the rapidly expanding cell

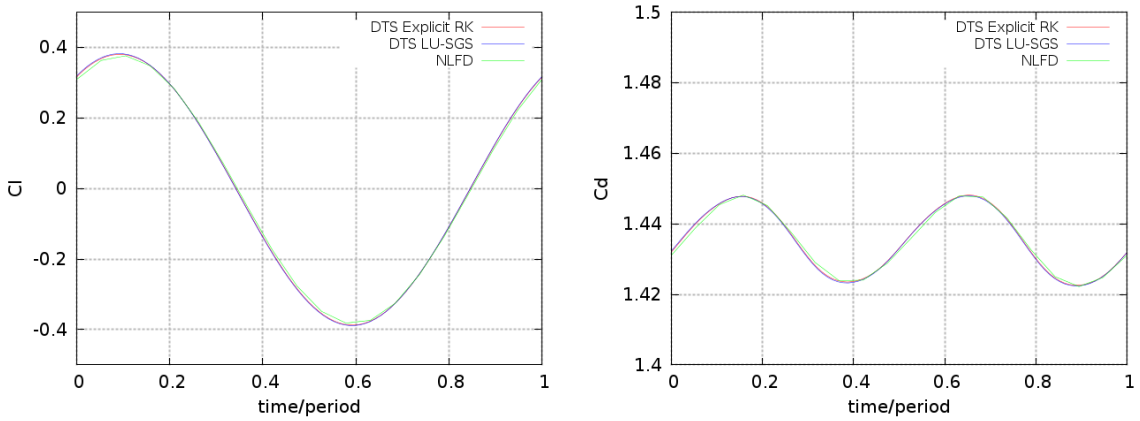


Figure 3.30 Lift and drag coefficient variations over a period

sizes in the wake which tend to dissipate vortices. To study the dissipation of vortices with a finer grid, a simulation using overset grids containing a refined background was done. The foreground mesh is a cropped version of the circular cylinder mesh used previously so that it only extends three diameters length outside the cylinder. The background mesh of size 256x256 contains rectangular cells, extends 50 diameters length in every directions from the cylinder and is refined near the center to provide better resolution of the vortices. The assembly of the overset grids can be seen on figure 3.31. Using these grids, vortices were observed up to twelve cylinder lengths away from the cylinder as can be observed in figure 3.32. This is an example of how overset grids can improve the quality of simulations.

A limitation in the usage of the NLFD method has also been encountered when simulating the flow around the circular cylinder. When increasing the number of modes used in the simulation, the stability of the solver seems to decrease. Figure 3.33 compares the convergence curves of the four modes resolution presented in the test case and an equivalent ten modes resolution. It is observed that while all modes used in the resolution converge when using four modes, most modes of the ten modes solution diverge.

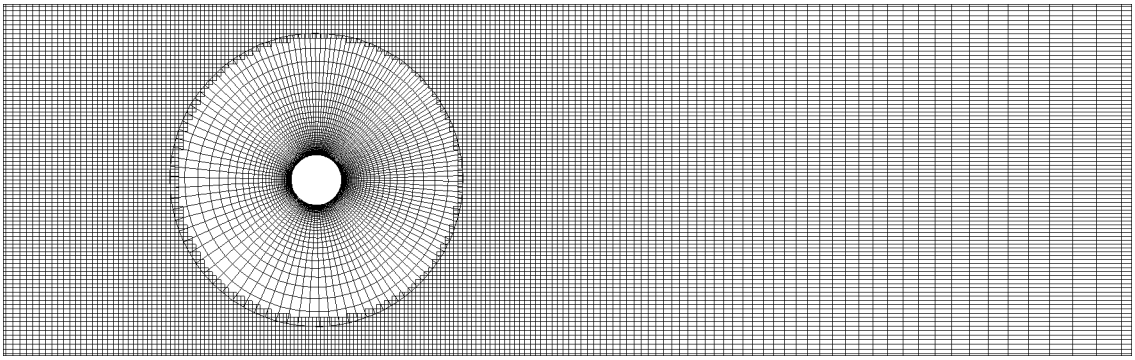


Figure 3.31 Overset mesh used on the circular cylinder simulation

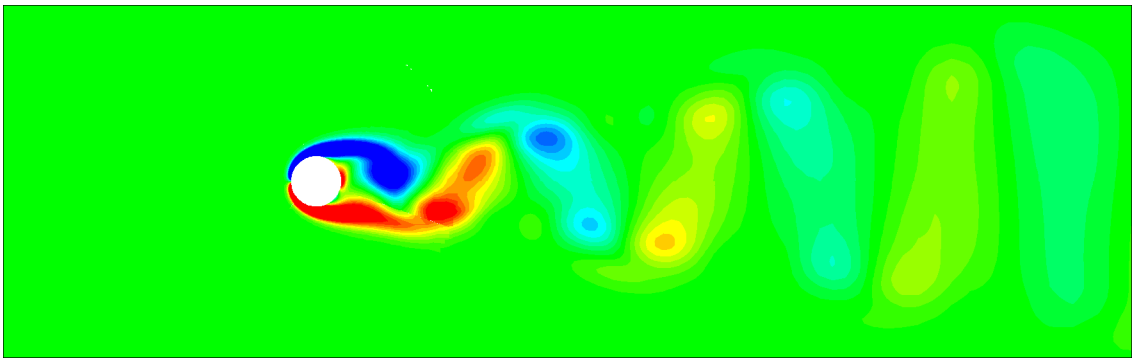


Figure 3.32 Vorticity contours on the circular cylinder simulation using an overset mesh

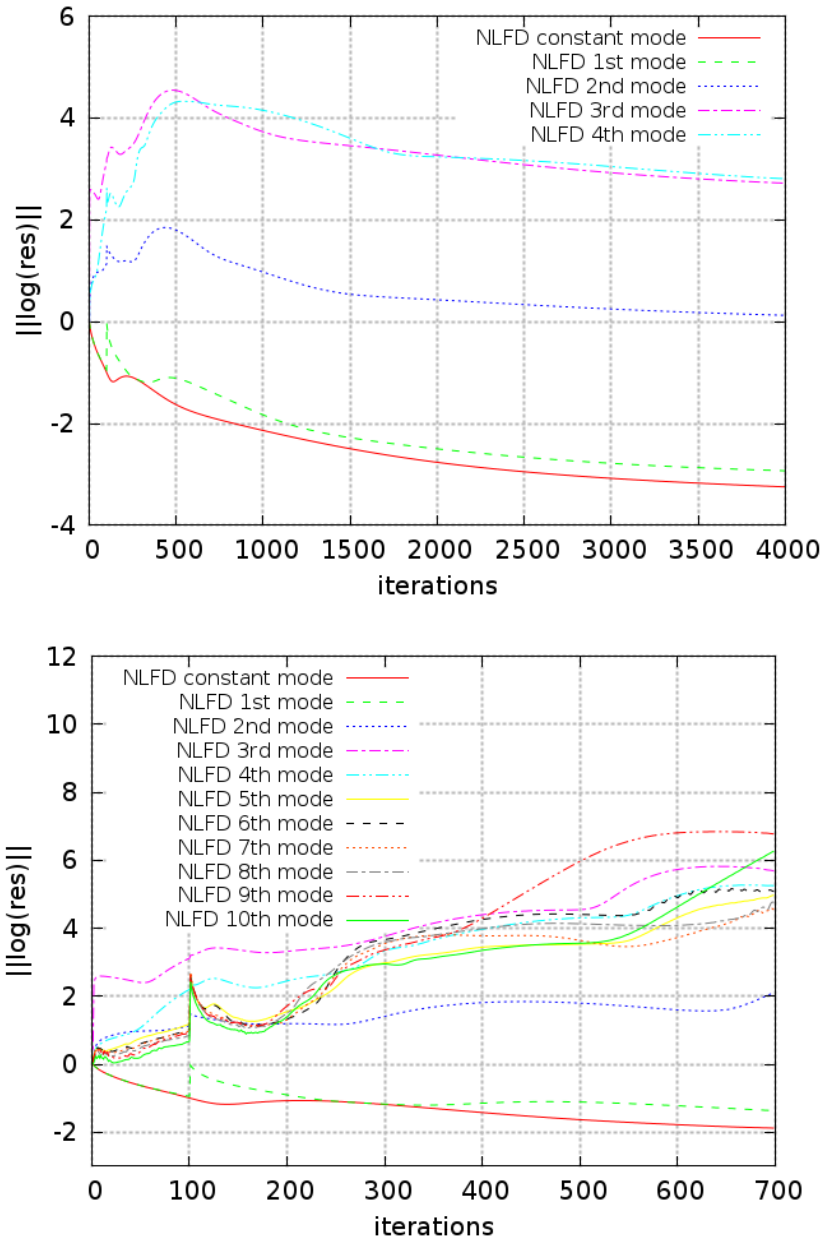


Figure 3.33 Convergence of each harmonic mode on a 4 modes (top) and 10 modes (bottom) NLFD resolution

CHAPTER 4 NUMERICAL RESULTS

This chapter applies the methods described earlier to application cases. Two particular cases are chosen to make best use of the methods:

- A high-lift airfoil optimisation;
- An unsteady simulation of turbulent flows in the frequency domain of a pitching airfoil.

The high-lift airfoil optimisation seeks to maximise the lift coefficient of the McDonnel Douglas Research airfoil by changing the flap's position using a two dimensional approach and a three dimensional approach with infinite swept wing hypothesis. The second case sought to introduce a turbulence component to the NLFD implementation in NSCODE. As a step before using one and two equations turbulence models, an algebraic turbulence model is used.

4.1 High-lift airfoil optimisation

The aim here is to optimise the high-lift MDA geometry for a landing configuration. To do so, the highest maximum lift coefficient is sought to allow for lower approach speeds. The optimisation uses a full mapping of the maximum lift coefficient of the geometry when moving the position of the flap in x and y direction. While requiring more computational resources, this method allows to capture the overall behaviour of the design space with respect to flap displacements. This problem was inspired by the work of Mavriplis and Mani.

The chimera method is used extensively in this study to modify the gap and overlap of the flap. In high-lift device vocabulary, the flap position variation in the x direction corresponds to a variation in the overlap while the variation in y direction corresponds to the gap between the main element and the flap. Figure 4.1 illustrates the concepts of gap and overlap between the flap and the main element, note that in this study, x and y variations correspond respectively to overlap and gap variation from the baseline MDA30N30P geometry. Since the mesh associated with each element is independent, no remeshing is needed to modify the geometry. Only the flap mesh has to be rigidly moved to the new flap position and the chimera preprocessing has to be redone to account for the new cells overlap. The chimera preprocessing can however be problematic in the tight gap between the flap and the main airfoil since the chimera boundary requires a depth of a few cells in order to ensure an accurate communication between the meshes. A large buffer, for example of three or more cells, could end up computing the flow on cells using next to or contained inside solid boundaries

which would yield inaccurate results. Interpolation errors also arise as the interpolated cells near the chimera boundaries will likely obtain their data from much smaller near wall cells of another mesh.

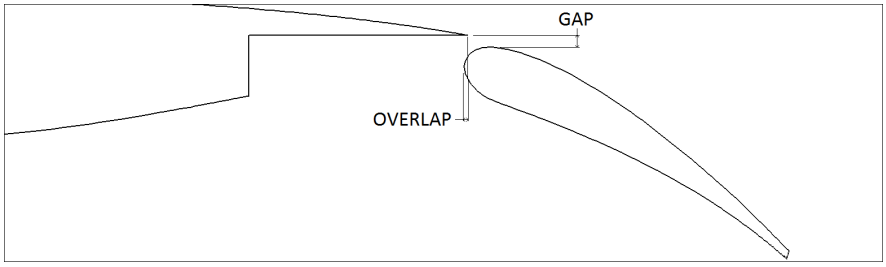


Figure 4.1 Gap and overlap definition for a flap

As a criteria, the smallest gap deemed acceptable for this analysis is determined as the smallest gap that allows two interpolated cells between solid wall boundaries and the computed region of a mesh. This limits the maximum y deflection of the flap to 0.01% of the chord length.

4.1.1 2.5D infinite swept wing

To show the importance of crossflow in high-lift aerodynamic design, the optimisation is conducted with unswept 2D and 30° swept wing conditions using the infinite-swept wing approach as developed by Ghasemi *et al.*. The two C_{Lmax} maps obtained are compared in order to assess the impact of sweep on high-lift design.

Figure 4.2 shows the difference in the C_L–α curve between the 2D resolution of the MDA airfoil without sweep and a 30° sweep infinite swept wing simulation. As expected, the $C_{L\alpha}$ ratio between the 30° swept wing and the purely 2D wing is almost cos(30°) between 4° and 8.1° angle of attack and would be of exactly cos(30°) in inviscid flow. The stall is also more abrupt and occurs at a smaller attack angle and lift coefficient in the swept wing simulation. These differences show that a 2D high-lift analysis and optimization without considering the effect of sweep leads to unrealistic multi-element designs.

To further illustrate the difference between a purely 2D flow and a 30° swept wing flow, figure 4.3 show the difference in flowfields at 21°angle of attack. As is showed by the abrupt stall characteristics of the infinite swept wing flow in figure 4.2, the main element and the flap are completely stalled at 21° angle of attack. The purely 2D resolution however show a flowfield where only a small portion of the trailing edge of the flap is stalled.

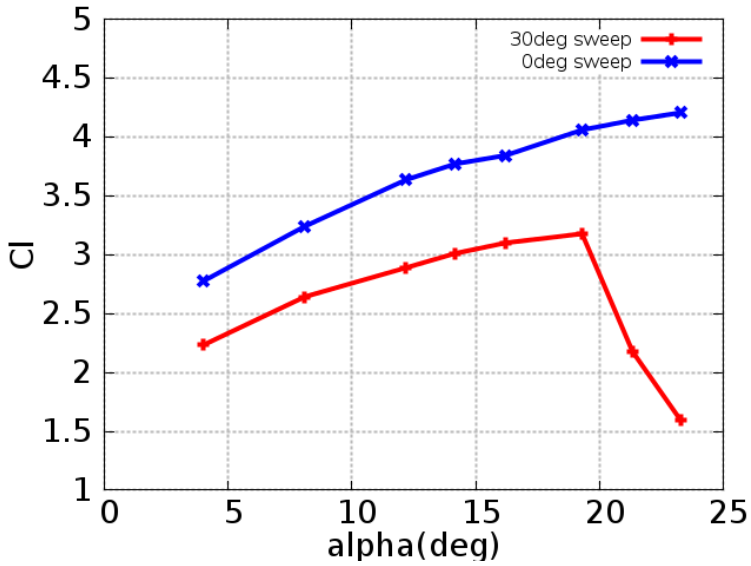


Figure 4.2 Cl–α curve of the MDA airfoil for a purely 2D simulation (0˚ sweep) and an infinite swept wing simulation with a sweep of 30˚

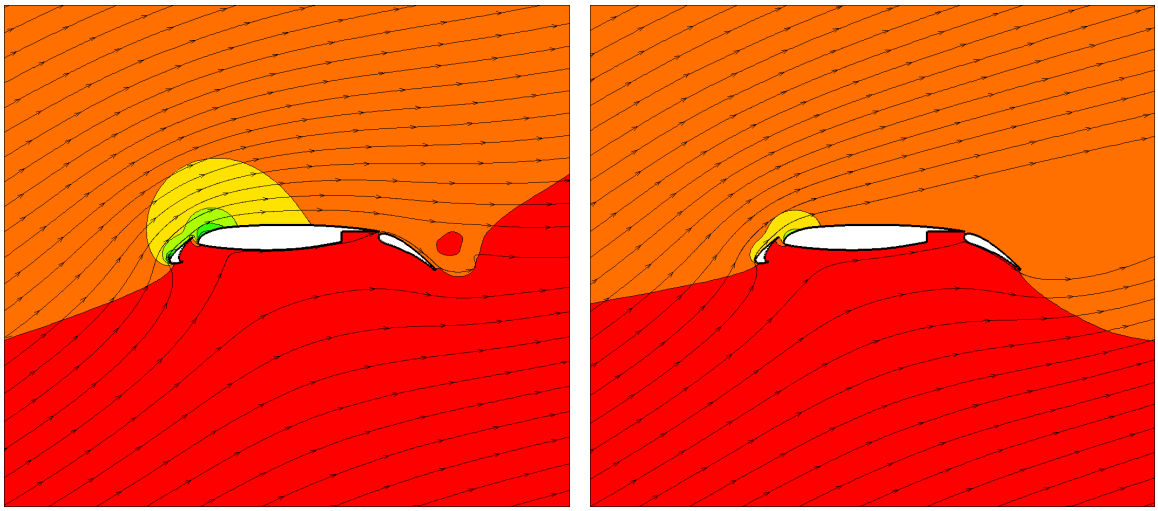


Figure 4.3 Streamlines and pressure field over a *MDA* geometry for a 2D (or infinite 0˚ swept wing) on the left and an infinite 30˚ swept wing on the right

4.1.2 Optimisation results

Figure 4.4 shows the results of the mapping for the 2D C_{Lmax} optimisation using purely 2D simulations on the left and 2.5D infinite swept wing approach on the right. As expected, the optimum point was obtained with the same parameters combinations from both mapping and the C_{Lmax} obtained was much lower with the infinite swept wing approach.

The study also shows that swept wings would be much more sensitive to parameters change. A notable effect captured by the 2.5D approach that is not observed in the purely 2D approach is the loss of performance at a translation of 0.015 or 1.5% of the flap in the x direction. This position roughly corresponds to the point were the flap no longer overlaps the main element.

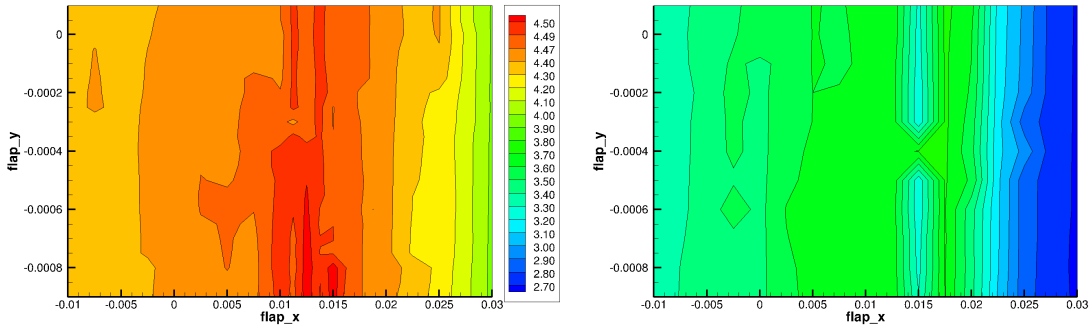


Figure 4.4 Lift coefficient obtained for each x and y displacement of the flap in a 2D flow on the left and an infinite 30° swept wing on the right

4.2 Turbulent NLFD case

4.2.1 Turbulence Model

The Baldwin Lomax turbulence model has been developped by Baldwin and Lomax (1978) to create an algebraic model for turbulence. The main reason leading to the choice of the Baldwin Lomax model here is to obtain turbulent results at low computational costs without extensive software changes. Since turbulence is currently computed decoupled from the conservative variables in the *NSCODE* framework, calculating the time derivative terms in turbulence models involving transport equation would require additional computational effort to recompute the Fourier transform of the conservative variables. The Baldwin Lomax turbulence model being algebraic, it does not require to solve transport equation or the Fourier coefficient of the conservative variables. It can directly be applied to the solution at each time sample of the NLFD solution without changing the NLFD solver in other ways.

4.2.2 Pitching NACA64A010 airfoil

The test case CT6 from Landon is chosen to test the turbulence model usage with NLFD simulations. The freestream Mach number is 0.796 and Reynolds number is 12.56x10^6. As mentionned previously, the Baldwin-Lomax turbulence model is used. The airfoil is a NACA64A010 profile undergoing a sinusoidal pitching motion around its quarter chord. The mean angle of attack of the motion is 0.00° with an amplitude of ±1.01°. The reduced frequency k_c is 0.202.

Figure 4.5 shows the "C" topology mesh used. The mesh has a size of 193x49, expands 20 chords from the domain and its wall distance has a y^+ of 1 or smaller on all the solid boundaries. The mesh parameters were selected to reproduce the study of Jameson *et al.* (2002). The mesh is again obtained with the *NSGRID* grid generator (Hasanzadeh *et al.*, 2013).

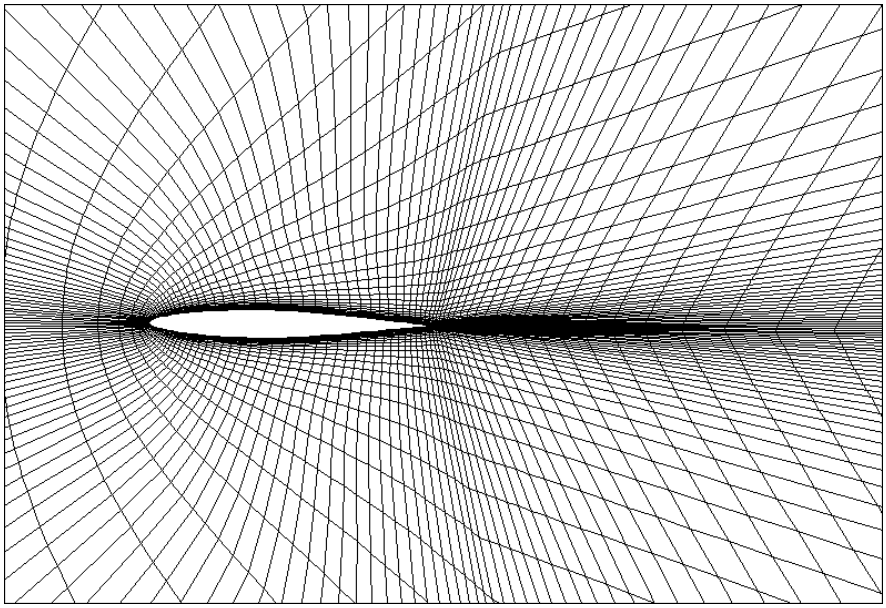


Figure 4.5 Mesh used for the turbulent pitching NACA64A010 test case

Figure 4.6 shows the lift and moment coefficient hysteresis versus the angle of attack of the airfoil. Good agreement is observed between the different numerical methods and the literature (Jameson *et al.*, 2002). Lift hysteresis is also in accordance to experimental results. As few as 2 NLFD modes are required to correctly evaluate the lift.

Figure 4.7 and 4.8 show a snapshot of the solution at 0.65° while the airfoil is moving in a nose down motion. The Mach contour and C_p distribution results are plotted using a NLFD resolution with 5 modes.

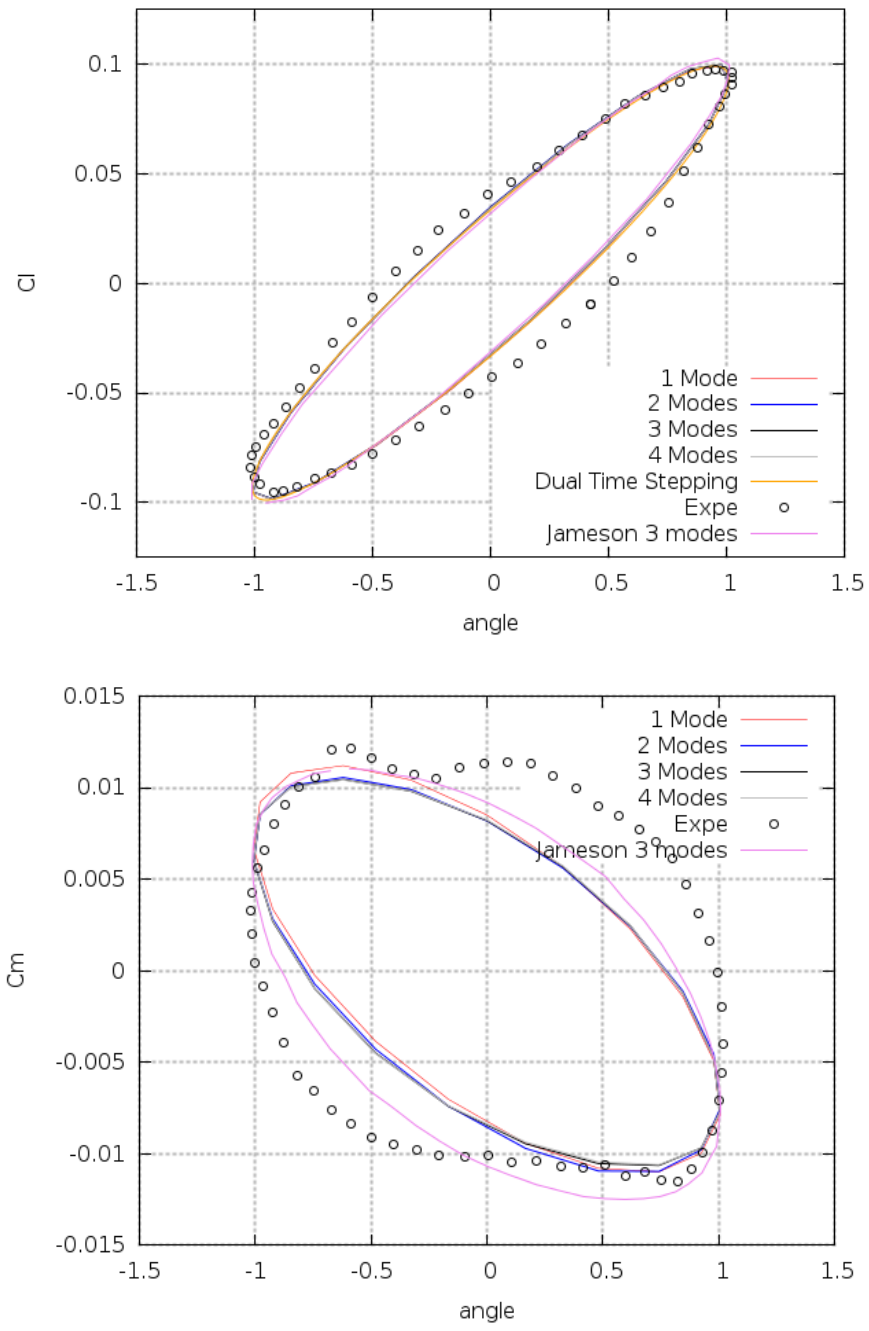


Figure 4.6 Lift and moment coefficients versus the angle of attack for the pitching turbulent NACA64A010 airfoil

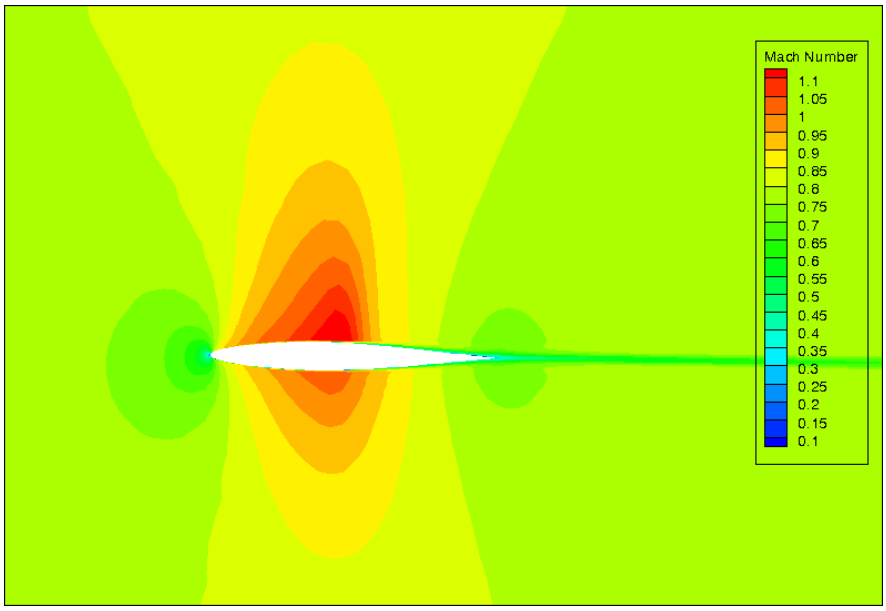


Figure 4.7 Mach contours at 0.65˚ in a nose down motion of the turbulent NACA64A010 airfoil

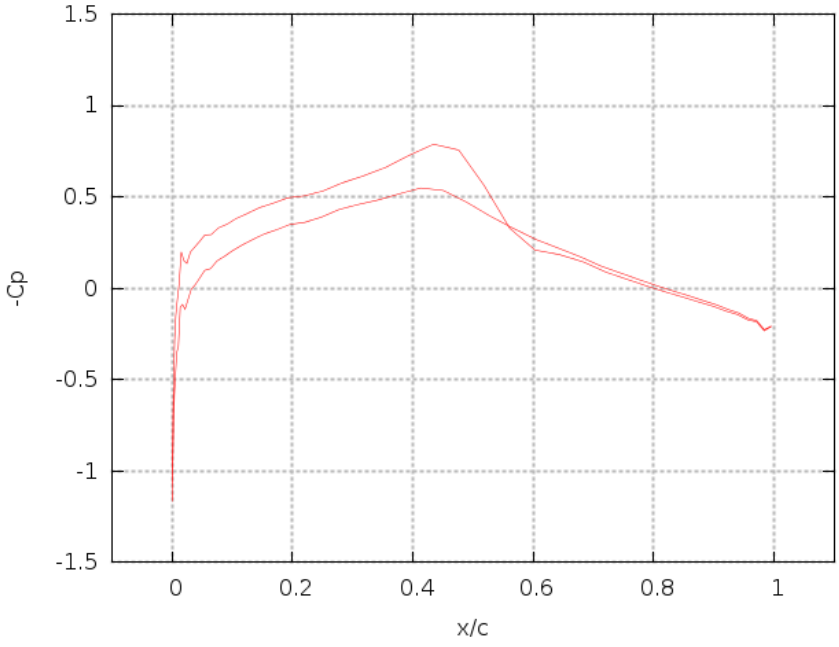


Figure 4.8 C_p distribution at 0.65˚ in a nose down motion of the turbulent NACA64A010 airfoil

CHAPTER 5 CONCLUSION

5.1 Synthesis of Work

This work presented in details the developments of a steady explicit RANS solver to incorporate various steady and unsteady solver schemes. Three main avenues have been explored to adress the current challenges of CFD.

Firstly, the ability of the software to handle complex geometries were improved using multi-block and overset grid technique. Not only did overset grids proved to be adequate to handle high-lift airfoils, but they were also used to improve the resolution of the wake of a circular cyclinder. Indeed, vortices dissipation was extended from four to twelve diameter using a cartesian background mesh refined in the wake region instead of a standard single block "O" type grid.

Secondly, a point implicit Jacobi preconditionner and an implicit LU-SGS scheme were implemented to improve the steady flow solver. Validation of these methods through various test cases showed the effectiveness of these developments. The Point Jacobi preconditionner proved to be inefficient for Euler cases but very good on RANS cases that had more stretched grids. The implicit LU-SGS scheme proved to be generally more robust in terms of convergence rate per solver iteration. However, the additionnal computational costs for each solver iteration generally balances the gains in convergence rates. The method is thus more robust at a comparable computationnal cost to explicit and preconditionned point implicit schemes. Every schemes were verified and validated with other CFD solvers and second order space accuracy was shown using a standard NACA0012 airfoil under inviscid flow conditions.

Thirdly, a Non Linear Frequency Domain solver was implemented in addition to the Dual Time-Stepping scheme to efficiently solve periodic unsteady flows. For the case tested, the NLFD method proved to yield the same results as the DTS scheme for periodic flows using somehow few harmonic modes in the simulation. The number of solver iterations to converge a NLFD simulation was shown to be similar to a steady simulation and the computationnal costs for each iteration scales linearly with the number of harmonic modes used. These properties of the NLFD solver make it much faster to obtain periodic solutions than a DTS simulation. The DTS time discretisation was verified to be of second order accuracy.

All these implementations were made with *NSCODE* using a carefully designed framework architecture. Low level laguage (C) is used for computationaly intensive calculations, while high-level language (*Python*) is used to control the various modules. The framework resides

inside a revision control system (*Mercurial*) to support the developpment of the code.

Using these developpments, two applications were performed

- Flap position optimisation to obtain the best possible C_{Lmax};
- Study of a turbulent NACA64A010 test case;

The flap optimisation study took advantage of the ability of the code to handle complex geometry and the improvements in steady solver to improve solution robustness. The study showed the optimal overlap position of the flap and the little influence of the gap on the C_{Lmax}. The benefit of using the infinite swept wing assumption was demonstrated. The turbulent pitching airfoil case showed the potential of using turbulence models coupled with the NLFD solver for efficient calculations of periodic flows.

5.2 Limitations of the Proposed Solution

Few limitations were encountered during software developments. First, the incompatibility of the chimera technique with multigrid acceleration technique is an important downside to the use of overset meshes. Indeed, as the multigrid method is one of the most effective techniques to obtain faster convergence of the flow solver, simulations using overset meshes are expected to be less performant.

A second limitation related to the use of overset grids is encountered when simulating multiple body geometries with narrow gaps between them. Using overset meshes, the required minimum rows of cells to ensure connections between grids of different bodies may not be respected, degrading the accuracy of the solution. Although Soucy and Nadarajah (2009) showed a technique to obtain residual convergence to machine accuracy using the multigrid technique on overset grids, the problem of coarse boundary definition on the coarse grids remains unsolved when dealing with multiple bodies close to one another.

Another limitation was observed on the NLFD implementation as the number of harmonic modes were increased, which is that a high number of modes in the computation tends to lead to slower convergence for each solver iterations and eventually divergence of the solution for very large number of modes. A possible solution has been proposed by McMullen (2003) with the *Coarse grid Spectral Viscosity* to adress this issue. Mundis and Mavriplis also recently showed promising results with using up to 47 modes with the time spectral method to solve an Euler problem.

Benchmark runs showed that *NSCODE* parallelised portion stands at 94.6% using Amdahl's

law, the maximum theoretical speedup is thus calculated at 18.5x. The parallel implementation was made with *OpenMP*, limiting the execution of the code on shared memory computers.

5.3 Future Work

In terms of complex geometry handling, patch grid treatment is currently investigated to handle overset grids on bodies intersecting each other. Patch grids are especially useful for wing-aileron configurations where there is no gap between the two bodies even though they are moving relative to one another.

Full implicit solvers, such as *GMRES* scheme, are currently considered to be implemented to further increase the convergence rate of *NSCODE*. Compatibility between implemented NLFD solver and LU-SGS scheme are also considered.

Several improvements in the NLFD method are possible in the near future. First, the Gradient Based Variable Time Period method (McMullen *et al.*, 2002) would make a fine addition to the NLFD solver in cases where the Strouhal number of the phenomenon is not known beforehand. This method enables the NLFD solver to adjust the Strouhal number of the simulation at each iteration of the solver in order to achieve machine accuracy when simulating unsteady periodic flows without forced motion. Another very interesting improvement currently investigated for the NLFD solver is to solve the transport equations of the conservative variables and of the turbulence models in a coupled manner. This modification would allow to decompose the turbulence models in the same fashion as the conservative variables and produce fully turbulent simulations using the NLFD solver and turbulence models such as Spalart-Allmaras or $k - \omega$.

Finally, the algorithms presented can all be used in aero-elastic problems, as well as extended to 3D flows.

REFERENCES

ALLMARAS, S. R., "Analysis of a local matrix preconditioner for the 2-d navier-stokes equations," in *11th Computational FLuid Dynamics Conference*. Orlando, FL: AIAA, 1993.

AREVALO, S., ATWOOD, C., BELL, P., BLACKER, T., DEY, S., FISHER, D., FISHER, D., GENALIS, P., GORSKI, J., HARRIS, A. *et al.*, "A new dod initiative: the computational research and engineering acquisition tools and environments (create) program," in *Journal of Physics: Conference Series*, vol. 125, no. 1. IOP Publishing, 2008, p. 012090.

BALDWIN, B. S. and LOMAX, H., *Thin layer approximation and algebraic model for separated turbulent flows*. American Institute of Aeronautics and Astronautics, 1978, vol. 257.

BLAZEK, J., *Computational Fluid Dynamics: Principles and Applications:(Book with accompanying CD)*. Elsevier, 2005.

CAGNONE, J., SERMEUS, K., NADARAJAH, S. K., and LAURENDEAU, E., "Implicit multigrid schemes for challenging aerodynamic simulations on block-structured grids," *Computers & Fluids*, vol. 44, no. 1, pp. 314–327, 2011.

DA RONCH, A., MCCRACKEN, A., BADCOCK, K., WIDHALM, M., and CAMPOBASSO, M., "Linear frequency domain and harmonic balance predictions of dynamic derivatives," *Journal of Aircraft*, vol. 50, no. 3, pp. 694–707, 2013.

DELOZE, T. and LAURENDEAU, E., "Nsmb contribution to the 2nd high lift prediction workshop," in *52nd Aerospace Sciences Meeting*, aIAA Paper 2014–0913.

FRIGO, M. and JOHNSON, S. G., "Fftw: An adaptive software architecture for the fft," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 3. IEEE, 1998, pp. 1381–1384.

GHASEMI, S., MOSAHEBI, A., and LAURENDEAU, E., "A two-dimensional/infinite swept wing navier-stokes solver," in *52nd Aerospace Sciences Meeting*, aIAA Paper 2014–0557.

HALL, K. C., THOMAS, J. P., and CLARK, W. S., "Computation of Unsteady Nonlinear Flows in Cascades Using Harmonic Balance Technique," *AIAA Journal*, vol. 40, no. 5, pp. 879–886, May 2002.

HASANZADEH, K., MOSAHEBI, A., LAURENDEAU, E., and PARASCHIVOIU, I., "Validation and verification of multi-steps icing calculation using canice2d-ns code," in *AIAA Fluid Dynamics and Co-located Conferences and Exhibit*. San Diego, California, USA: AIAA, 2013.

HIRSCH, C., *Numerical Computation of Internal and External Flows: The Fundamentals of Computational Fluid Dynamics: The Fundamentals of Computational Fluid Dynamics*. Butterworth-Heinemann, 2007, vol. 1.

JAMESON, A., "Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings," in *AIAA 10th Computational Fluid Dynamics Conference*, vol. AIAA 91-1596, Honolulu, HI, 1991.

JAMESON, A., SCHMIDT, W., and TURKEL, E., "Numerical solution of the euler equations by finite volume methods using runge-kutta time-stepping schemes," *AIAA paper*, vol. 1259, 1981.

JAMESON, A., "Solution of the euler equations for two dimensional transonic flow by a multigrid method," *Applied mathematics and computation*, vol. 13, no. 3, pp. 327–355, 1983.

JAMESON, A., "Multigrid algorithms for compressible flow calculations," in *Multigrid Methods II*. Springer, 1986, pp. 166–201.

JAMESON, A. and BAKER, T. J., "Solution of the euler equations for complex configurations," *AIAA paper*, vol. 83, 1983.

JAMESON, A. and YOON, S., "Lower-upper implicit schemes with multiple grids for the euler equations," vol. 25, no. 7, 1987, pp. 929–935.

JAMESON, A., ALONSO, J., and MCMULLEN, M., "Application of a non-linear frequency domain solver to the euler and navier–stokes equations," *AIAA paper*, vol. 13625, 2002.

LANDON, R., "Naca 0012 oscillating and transient pitching, data set 3," *Compendium of unsteady aerodynamic measurement, AGARD*.

LEVESQUE, A. T., PIGEON, A., DELOZE, T., and LAURENDEAU, E., "An overset grid 2d/infinite swept wing urans solver using recursive cartesian virtual grid method," in *53RD AIAA AEROSPACE SCIENCES MEETING*, aIAA Paper 2015–0912.

LIAO, W., CAI, J., and TSAI, H. M., "A multigrid overset grid flow solver with implicit hole cutting method," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 9–12, pp. 1701–1715, 2007.

MARTINELLI, L., "Calculation of viscous flows with a multigrid method," Ph.D. dissertation, Dept. of Mechanical and Aerospace Engineering, Princeton University, NJ., 01 1987.

MAVRIPLIS, D. J. and MANI, K., "Adjoint-based shape optimisation of high-lift airfoils using the nsu2d unstructured mesh solver," in *52nd Aerospace Sciences Meeting*, aIAA Paper 2014–0554.

MCMULLEN, M., JAMESON, A., and ALONSO, J., "Acceleration of Convergence to a Periodic Steady State in Turbomachinery Flows," in *39th AIAA Aerospace Sciences Meeting & Exhibit*, vol. AIAA 2001-0152, Reno, NV, 2001.

MCMULLEN, M., JAMESON, A., and ALONSO, J., "Application of a Non-Linear Frequency Domain Solver to the Euler and Navier-Stokes Equations," in *40th AIAA Aerospace Sciences Meeting & Exhibit*, vol. AIAA 2002-0120, Reno, NV, 2002.

MCMULLEN, M. S., "The application of non-linear frequency domain methods to the euler and navier-stokes equations," Ph.D. dissertation, Citeseer, 2003.

MORINISHI, K., "A finite difference solution of the euler equations on non-body-fitted cartesian grids," *Computers & Fluids*, vol. 21, no. 3, pp. 331–344, 1992.

MOSAHEBI, A. and NADARAJAH, S., "An adaptative Non-Linear Frequency Domain method for viscous flows," *Computers & Fluids*, vol. 75, pp. 140–154, 2013.

MUNDIS, N. L. and MAVRIPLIS, D. J., "An efficient flexible gmres solver for the fully-coupled time-spectral aeroelastic system."

PIERCE, N. and GILES, B., "Preconditioned multigrid methods for compressible flow calculations on streched meshes," *Journal of Computational Physics*, vol. 136, no. CP975772, pp. 425–445, 1997.

PIGEON, A., "Accélération de la résolution des équations de type urans sur des ailes d'avions transsoniques," Master's thesis, Polytechnique Montreal, QC, CANADA, 2015.

PIGEON, A., LEVESQUE, A. T., and LAURENDEAU, E., "Two-dimensional navier-stokes flow solver developments at École polytechnique de montréal," in *CFD Society of Canada 22nd Annual Conference*. Toronto, CA: CFDsc, 2014.

PULLIAM, T. H. and STEGER, J. L., "Recent improvements in efficiency, accuracy, and convergence for implicit approximate factorization algorithms," *AIAA paper*, vol. 854360, 1985.

ROE, P., "Approximate riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, vol. 43, pp. 357–372, 1981.

RUMSEY, C. L., SLOTNICK, J., LONG, M., STUEVER, R. A., and WAYMAN, T. R., "Summary of the first aiaa cfd high-lift prediction workshop," *Journal of Aircraft*, vol. 48, no. 6, pp. 2068–2079, 2011.

SHAROV, D. and NAKAHASHI, K., "Reordering of 3-d hybrid unstructured grids for vectorized lu-sgs navier-stokes computations," *AIAA paper*, vol. 97, pp. 2102–2117, 1997.

SONI, K., CHANDAR, D. D. J., and SITARAMAN, J., "Development of an overset grid computational fluid dynamics solver on graphical processing units," *Computers & Fluids*, vol. 58, no. 58, pp. 1–14, 2012.

SOUCY, O. and NADARAJAH, S. K., "A nlfd method for the simulation of periodic unsteady flows for overset meshes." in *19th AIAA Computational Fluid Dynamics*, 2009.

SOUTH JR, J. C. and BRANDT, A., "Application of a multi-level grid method to transonic flow calculations," 1976.

SUHS, N. E., ROGERS, S. E., DIETZ, W. E., and KWAK, D., "Pegasus 5: An automated pre-processor for overset-grid cfd," 2002.

SWANSON, R. and TURKEL, E., "On some numerical dissipation schemes," *Journal of Computational Physics*, vol. 147, no. CP986100, pp. 518–544, 1998.

SWANSON, R. R. and TURKEL, E., "Multistage schemes with multigrid for euler and navier-stokes equations," *NASA Technical Paper*, no. 3631, 1997.

THIBERT, J. J., "Cfd platforms: an introduction to onera's softwares."

VASSBERG, J. C. and JAMESON, A., "In pursuit of grid convergence for two-dimensional euler solutions," *AIAA Paper*, no. 81-1259, pp. 1152–1166, 1981.

VERSTEEG, H. K. and MALALASEKERA, W., *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education, 2007.

WANG, Z. J. and PARTHASARATHY, V., "A fully automated chimera methodology for multiple moving body problems," *International Journal for Numerical Methods in Fluids*, vol. 33, no. 7, pp. 919–938, 2000.

WEBER, C., "Developpement de methodes implicites pour les equations de navier stokes moyennees et la simulation des grandes echelles: Application al aerodynamique externe," *These de doctorat, INPT Toulouse*, 1998.

WILLIAMSON, C., "Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low reynolds numbers," *Journal of Fluid Mechanics*, vol. 206, pp. 579–627, 1989.

WRIGHT, M. J., CANDLER, G. V., and PRAMPOLINI, M., "Data-parallel lower-upper relaxation method for the navier-stokes equations," *AIAA journal*, vol. 34, no. 7, pp. 1371–1377, 1996.

YOON, S. and JAMESON, A., *An Multigrid LU-SSOR Scheme for Approximate Newton Iteration Applied to the Euler Equations*. National Aeronautics and Space Administration, 1986.

YOON, S. and JAMESON, A., "Lower-upper symmetric-gauss-seidel method for the euler and navier-stokes equations," *AIAA journal*, vol. 26, no. 9, pp. 1025–1026, 1988.

APPENDIX A TOPOLOGY FILE EXAMPLE

This annex contains the structure of a topology input file of *NSCODE*. It is an example for a mesh around a NACA0012 airfoil. The mesh was initially a single block "O" mesh divided into four equal blocks. The mesh size was initially 65 by 65 nodes.

NACA0012					
4					
17	65				
17	65				
17	65				
17	65				
BC	is	ie	js	je	icomp
block#	1		nsubface	4	
CON	1	1	1	65	0
4	17	17	1	65	0
CON	17	17	1	65	0
2	1	1	1	65	0
WAL	1	17	1	1	1
FAR	1	17	65	65	0
block#	2		nsubface	4	
CON	1	1	1	65	0
1	17	17	1	65	0
CON	17	17	1	65	0
3	1	1	1	65	0
WAL	1	17	1	1	1
FAR	1	17	65	65	0
block#	3		nsubface	4	
CON	1	1	1	65	0
2	17	17	1	65	0
CON	17	17	1	65	0
4	1	1	1	65	0
WAL	1	17	1	1	1
FAR	1	17	65	65	0
block#	4		nsubface	4	
CON	1	1	1	65	0
3	17	17	1	65	0
CON	17	17	1	65	0
1	1	1	1	65	0
WAL	1	17	1	1	1
FAR	1	17	65	65	0

APPENDIX B VISCOUS JACOBIANS

The current Appendix formulates the different viscous flux Jacobians used throughout the CFD community. For simplification purposes, only 2D Jacobians in the I computationnal direction will be written.

The first formulation is from Blazek (2005) and uses the Thin Shear Layer approximation.

$$\bar{A}_v = \frac{\mu}{\Omega}\begin{bmatrix} 0 & 0 & 0 & 0 \\ b_{21} & a_1\partial_I\left(\rho^{-1}\right) & a_2\partial_I\left(\rho^{-1}\right) & 0 \\ b_{31} & a_2\partial_I\left(\rho^{-1}\right) & a_3\partial_I\left(\rho^{-1}\right) & 0 \\ b_{41} & b_{42} & b_{43} & a_4\partial_I\left(\rho^{-1}\right) \end{bmatrix} \tag{B.1}$$

where,

$$\begin{aligned} \partial_I(\phi) &= \frac{\partial\phi}{\partial I} \\ a_1 &= \frac{4}{3}I_x^2 + I_y^2 \\ a_2 &= \frac{1}{3}I_xI_y \\ a_3 &= I_x^2 + \frac{4}{3}I_y^2 \\ a_4 &= \left(\frac{\gamma}{Pr}\right)\left(I_x^2 + I_y^2\right) \\ b_{21} &= -a_1\partial_I\left(\frac{u}{\rho}\right) - a_2\partial_I\left(\frac{v}{\rho}\right) \\ b_{31} &= -a_2\partial_I\left(\frac{u}{\rho}\right) - a_3\partial_I\left(\frac{v}{\rho}\right) \\ b_{41} &= -a_4\partial_I\left(\frac{(u^2+v^2)}{\rho} - \frac{E}{\rho}\right) - a_1\partial_I\left(\frac{u^2}{\rho}\right) \\ -2a_2\partial_I\left(\frac{uv}{\rho}\right) - a_3\partial_I\left(\frac{v^2}{\rho}\right) & \\ b_{42} &= -a_4\partial_I\left(\frac{u}{\rho}\right) - b_{21} \\ b_{43} &= -a_4\partial_I\left(\frac{v}{\rho}\right) - b_{31} \end{aligned} \tag{B.2}$$

The second formulation from Sharov and Nakahashi (1997) consists of taking a diagonal matrix with the viscous spectral radius on the diagonal elements.

$$\bar{A}_v = \frac{\mu}{\Omega}\begin{bmatrix} \lambda^I_{visc} & 0 & 0 & 0 \\ 0 & \lambda^I_{visc} & 0 & 0 \\ 0 & 0 & \lambda^I_{visc} & 0 \\ 0 & 0 & 0 & \lambda^I_{visc} \end{bmatrix} \tag{B.3}$$

with λ^I_{visc} the viscous spectral radius in the I direction formulated as (Blazek, 2005)

$$\lambda^I_{visc} = max\left(\frac{4}{3\rho}, \frac{\gamma}{\rho}\right)\left(\frac{\mu_L}{Pr_L} + \frac{\mu_T}{Pr_T}\right)\frac{\left(\Delta S^I\right)^2}{\Omega} \tag{B.4}$$

Pr_L and Pr_T are the Prandtl numbers using respectively the laminar and turbulent viscosity.

The third formulation comes from the European *NSMB* flow solver (Weber, 1998). The viscous Jacobian is decomposed into its cartesian components

$$\bar{B}^x_v = \frac{\partial F_v}{\partial U_x} \quad , \quad \bar{B}^y_v = \frac{\partial F_v}{\partial U_y} \tag{B.5}$$

With U the primitive variables

$$\bar{B}^x_v = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{4}{3}\mu s_x & \mu s_y & 0 \\ 0 & -\frac{2}{3}\mu s_y & \mu s_x & 0 \\ -\frac{kTs_x}{\rho} & \frac{2}{3}\mu\left(2us_x - vs_y\right) & \mu\left(vs_x - us_y\right) & \frac{kTs_x}{p} \end{bmatrix} \tag{B.6}$$

$$\bar{B}^y_v = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \mu s_y & -\frac{2}{3}\mu s_x & 0 \\ 0 & \mu s_x & \frac{4}{3}\mu s_y & 0 \\ -\frac{kTs_y}{\rho} & \mu\left(vs_x - us_y\right) & \frac{2}{3}\mu\left(-us_x + 2vs_y\right) & \frac{kTs_y}{p} \end{bmatrix} \tag{B.7}$$

The Jacobians must then be transformed to the conservative variables formulation before being assembled

$$\bar{A}^\psi_v = \bar{B}^\psi_v \cdot \frac{\partial W}{\partial U} \tag{B.8}$$

$$\bar{A}_v = \Omega\left(n_x\bar{A}^x_v + n_y\bar{A}^y_v\right) \tag{B.9}$$

With Ω being the surface of the cell face.